ZX PRO/FILE

A Machine Language Data Base For The Timex

Thomas B. Woods

(c) 1983 Thomas B. Woods
all rights reserved

This program is dedicated to all those Timex and ZX81 owners who refuse to believe the Apple, TRS-80, OSI, IBM, Commodore, and Atari users and dealers who constantly remind us that the Timex 1000 "is just a toy".

ZX PRO/FILE

by

Thomas B. Woods

A file and text storage tool
for Timex and ZX81 computers

Complete how-to-use instructions
and program listings

CONTENTS

INTRODUCTION TO ZX PRO/FILE

ZX PRO/FILE is a record storage tool for your Timex Sinclair computer. You can use it for names, addresses, sales records, reference materials, charts and tables, statistics, research data, virtually any piece of important information.

Unlike other programs dedicated to the task of data storage and retrieval, ZX PRO/FILE does not limit you to one set file size. You can keep both short and long files in the same program without wasting memory space. What's more, you can store a lot if information. With 16K of RAM, you can store just under 11,000 characters configured in files ranging from just one to over 300 characters in length.

If you use a printer with your computer, ZX PRO/FILE offers tremendous versatility in printing large amounts of data. Long lists of information are possible, and you can define just what part of a file gets printed- from one line to any group of lines.

ZX PRO/FILE uses a lot of fast machine language programming to provide lightning quick access to any file you need. There are several search modes that let you select any record or group of records stored in memory.

LOADING THE ZX PRO/FILE CASSETTE

ZX PRO/FILE is loaded into your computer as any other program would be. Put the tape in your recorder, connect the cables, set the volume at its usual setting and type:

LOAD "ZX"

Turn the tape recorder on and push ENTER. Loading time is approximately six minutes. If ZX PRO/FILE fails to load the first time, try a slightly different volume setting. Also, re-check the cables, and if your recorder has a tone setting, adjust it so that bass is at a minimum.

Past experience indicates to me, at least, that 90% of all loading problems are due to the tape recorder rather than the computer or tape. Occasionally the tape head alignment will be off ever-so-slightly from that of the equipment that made the tape, resulting in a no-load situation. Other times, volume will not be high enough even at the maximum setting, or even worse, "noise" from the tape player will cause system failure.

If you encounter difficulties that won't go away, try another tape recorder before you send the cassette in for replacement. It may make all the difference.

Assuming everything loads ok, ZX PRO/FILE will self start. The TV screen will fill with what is called the Main Add/Search Menu.

THE MAIN ADD/SEARCH MENU

Here is what the Main Menu will look like:

```
ZX PRO/FILE

ENTER A SEARCH COMMAND
OR TYPE "A" TO ADD
      "SAVE" TO SAVE

SPACE OPEN: 10980 SLOTS


SEPARATE MULTI-WORD
COMMANDS WITH A "/"

TYPE "AUTO" FOR AUTO-SEARCH
     "DEFP"  FOR PRINT FORMAT


  PRINT STATUS
START/NO./SPACE
   1    11     1
```

The Main Menu tells you a lot about the state the computer is in. All of the information displayed on the screen will be covered in greater detail later in the manual, but briefly, the Main Menu tells you:

A) To enter a SEARCH COMMAND. This is how you will look information up after you start using the program. You can type any word, symbol, or phrase. The computer will then scan the files and print the files that contain the word you input as a search command.

B) TYPE "A" TO ADD. This puts the computer in the ADD/EDIT mode. Any time you want to add a new file, type "A".

C) Type "SAVE" to SAVE. Every time you add new files or change old ones, your cassette goes out of date. You must make new copies periodically to reflect the changes in the files you store. When you type SAVE, the computer automatically SAVES itself and then returns to the Main Menu.

D) SPACE OPEN: is an indicator of how much memory is filled with files and how much remains open. If you use the 16K memory pack, you can store 10980 characters configured in files of any size you want. The SPACE OPEN indicator grows smaller as you add more information. WARNING: Pay attention to the amount of space available to you. As the number of slots approaches zero, begin deleting old obsolete files to make more room. If you try to put more data into the computer than space allows, the program will go berzerk!

E) SEPARATE MULTI-WORD COMMANDS WITH A "/".
ZX PRO/FILE can search for files on the basis of the words they contain. A search command can be one word or two separate words. For example, you could tell the computer to list all files which contain a specific word, or to list only those files which contain both word A and word B. When you input such a multi-word command, you must separate the words with a "/". When the ZX PRO/FILE sees this character, it knows that there are two separate words, symbols, or phrases it must search for.

F) Type "AUTO" for AUTO-SEARCH. The AUTO-SEARCH function is used for producing a display of files in an ordered manner. The basis upon which files are ordered is on numerical data you place in the last word of any given file. Also, the AUTO-SEARCH function can be used with a compatible printer to give a complete list of all files containing the search command in question. This is useful for mailing lists printed on labels, for example. ZX PRO/FILE is designed to work with the Memotech Centronics I/F parallel interface. See the chapter on "Program Modifications" if you use the Sinclair or Timex printer.

G) Type "DEFP" for Print Format. DEFP stands for "Define Printer". Initially, the printer format is set to print the maximum of eleven lines of any given file with one space between it and the next file printout. This printer format is completely definable by you which is what you set when you type DEFP. You select the starting line to be printed, the total number of lines printed, and the number of spaces to line-feed between forms.

H) PRINT STATUS is shown at the bottom of the Main Menu. It tells you the current Print format. START/NO./SPACE represent the starting line to print, the number of lines to print, and the number of spaces to skip before printing the next file. The values given remain fixed until you change them by typing DEFP.

****

ADDING FILES TO ZX PRO/FILE

When you type "A" from the Main Menu, you put the computer in the ADD/EDIT mode. The screen clears and new instructions are displayed. At the top of the TV screen you will see a blinking ">" and immediately to its right is an asterisk. This asterisk is the first character of the first line of your new file. The blinking ">", called the Line Select Cursor, points to the file line at which you will add, alter, or delete information.

A prompt at the bottom of the screen says:

PRESS "C" TO CLOSE THE FILE
ARROWS MOVE THE ">"
HIT ENTER TO INPUT DATA

Press the keys that have the up or down pointing arrows on them (the 6 or 7 keys) to move the cursor up or down. When the cursor is "pointing" to the line you wish to input, hit ENTER to fix the cursor and initiate the input of the file line.

Since you haven't added any files yet, move the cursor so that it points to the top line, then press ENTER. The cursor stops blinking and the computer stops to let you input a line of text. As an example, try typing:

SOFTWARE then ENTER.

When you do, the word "SOFTWARE" gets printed on the TV screen. The asterisk is still the first character. This symbol is very important to ZX PRO/FILE. It tells the computer that this line is the beginning of a new file. Every file held in the program will begin with the "*".

After you enter the first line of your file, the cursor starts blinking again, and its position is advanced to the next line down. At any time the arrow keys will move the cursor to a different position, but for now keep it where it is. Press ENTER again to stop the blinking and input the second file line:

THOMAS B. WOODS then ENTER.

As before, the routine repeates. Using the same "Press ENTER/Input Line/ Press Enter" technique, finish adding these lines:

P.O. BOX 64 then
JEFFERSON, NH 03583

Now you have my name and address ready to file. When the cursor is blinking, press "C" to close the file. The screen goes blank for a brief period, and then you return to the Main Add/Search Menu. THOMAS B. WOODS is safely locked in the computer.

All files are added to ZX PRO/FILE in this way. Here are some additional notes and tricks to help you add files constructively. Also read about EDITING files because these two functions are almost identical.

FILE ADDING NOTES

Notice that when you press the arrow to move the cursor down, you can "skip" lines on which you input data. This feature gives you the ability to leave blank lines between printed ones. If you want to insert a new line between two old ones, just leave some space.

However, when you CLOSE the file, all blank lines are automatically deleted. If you need to leave permanent blank lines, type just one character such as a period on the line. Then, when you want to fill in the line with text, you can later EDIT the single character line with your new data.

One rule you must follow: Do not begin a file line with a blank space. If you do, the computer will treat the line as if it were completely empty, and thus delete it when you close the file. Spaces can occupy any other portion of the file without affecting program operation. Remember, to reserve blank lines for later insertions, put in at least one character on the line.

Line length is limited to 28 characters. The asterisk on the first line counts as one character. If you try to type more. the computer rejects the data and asks you to re-enter a shorter line.

If, after entering a line of a file, you see that you made a typographical error, you can change the line by pressing ENTER when the Line Select Cursor is pointing to the line in error. When the computer asks you to input data, type in the correct line.

Similarly, lines can be deleted. Stop the cursor at the line you want to erase. Then when ZX PRO/FILE asks you to input data, press just ENTER. What happens in this case is that you are actually inputting data. It's just that the data is nothing! Consequently, the line gets changed to nothing.

Any line of a file can be deleted except the first. If you delete the top line of a file, the entire file gets deleted. A return to the Main Menu is then made.

More file adding notes can be found in the chapter "Making ZX PRO/FILE Work for You".

Add/File Summary

1. Type "A" from the Main Menu.
2. After the Line Select Cursor appears, press ENTER.
3. Input your line of text.
4. If necessary, move the cursor by pressing the up/down arrows.
5. Repeat steps 2-4 until you are ready to close the file.
6. Press "C" when the cursor is blinking to CLOSE and return to the Main Add/Search Menu.

Remember: Lines are limited to 28 characters. Don't delete the top line unless you want to delete everything, and don't begin a file line with a blank space.

When you return to the Main Menu, you'll notice that the SPACE OPEN indicator is less than it was initially. Every time you change your files, SPACE OPEN reflects the new amount of space left for more data. Don't forget to heed this indicator. IF YOU TRY TO ADD MORE FILES THAN SPACE ALLOWS, PROGRAM FAILURE WILL RESULT.

GETTING DATA OUT

Once you have data stored in ZX PRO/FILE, there are many possible ways to get it out. All of them involve inputting a symbol, word, or phrase and having the computer follow the mundane and typically human course of searching every file stored in memory until a match to the word is found. Then the file is printed.

What makes the computer so much better than its human counterparts is its blinding speed. Even when the last available memory slot is taken, ZX PRO/FILE can search, find, and display any file in memory in less than a second!

Search Command Words

A Search Command is your acess to the files stored in ZX PRO/FILE. This is a word, symbol, or phrase that you know (or think) is held in the file you want to look up. It can be up to 28 characters in length (one complete file line).

Once entered, each file which contains the same characters will be displayed one by one. If you entered my name and address as a file:

```
*SOFTWARE
THOMAS B. WOODS
P.O. BOX 64
JEFFERSON, NH 03583
```

you could get me on the TV screen by inputting search commands like:

```
THOMAS B. WOODS
THOMAS
WOODS
B. WO
03583
X 64
NH
*
```

Any character or group of characters--as long as they're in the right order--will display the file in which they are found.

If you use ZX PRO/FILE as a mailing list, you could enter "NH" as a Search Command to list every name and address from "NH". Similarly, a search for SMITH would result in a display of every Mr./Mrs. SMITH filed. This kind of search is called a SINGLE WORD SEARCH even

though you may be looking for just one or two characters, or a whole series of characters which comprise a phrase like JEFFERSON, NH 03583.

Remember that every file begins with an asterisk (*). You can input this symbol as a Search Command to give you a display of every file held in the program. A search for "*" means in English: PRINT ALL FILES.

There are a few words you can't use as Search Commands because they initiate special instructions from the Main Menu. They are: "/", "AUTO", "SAVE", "DEFP", and "A".

Multi-Word Searches

ZX PRO/FILE can do more than search for just one word. It can also display files that hold two separate and distinct words. In a single word search, you tell the computer fo find all files that contain word A. The multi-word search tells the computer to select only those files that contain both word A and word B.

Multi-word commands are entered the same way that single word commands are, but you must put a "/" between the two words. For example,

WOODS/NH

entered as a search command would display only those people in your mailing list named WOODS if they live in NH. Smiths from NH or Woods from California would be rejected. If you use ZX PRO/FILE to record your collection of wild flower slides, a multi-word search for:

5 PETALS/RED

would display only those records of slides of red 5 petaled blossoms (assuming you had this information in your computer). All other flowers with 5 petals, or those with colors of different shades would be passed over.

A single word command lists all pertinent files; a multi-word command selects.

THE DISPLAY OPTION MENU
(or now that I found the file, what do you want me to do with it?)

Regardless of the number of words you search for, ZX PRO/FILE scans the data until it comes to the first match it finds. Then it prints the file. It is quite possible that other files also contain matches to the Search Command. The computer has no way of knowing if the file it finds is the one you want.

After every display of a found file, THE DISPLAY OPTION MENU is printed at the bottom of the TV screen. It reads:

HIT ENTER TO CONTINUE SEARCHING
"C" TO COPY
"R" TO RETURN TO PREVIOUS FILES
"E" TO EDIT THIS FILE
"N" TO BEGIN A NEW FILE SEARCH

Your choice tells the computer what to do next.

ENTER
If you press ENTER, the program continues its search for more matches. It picks up where it left off and displays any other files of the same Search Command.

Searching continues until the entire block of data has been checked. Then ZX PRO/FILE displays:

SEARCH IS COMPLETE

"C" TO COPY
Press "C" to send the file out to your printer. The parameters you set when you type DEFP from the Main Menu are in effect. If you told the computer to print lines 3 to 8 with one space between forms, this part of the displayed file is all that will be printed--even if all of the 12 maximum file lines hold information.

After the file is printed, you return to the same file display. You can press "C" again to make a second copy, or you can select another Display Option.

Note: ZX PRO/FILE is designed to work with the Memotech Centronics interface. It will not work with Timex printers unless the program is altered slightly. Instructions are provided in "Program Modifications".

"R" TO RETURN TO PREVIOUS FILES
If you use the program to store names and addresses, and you are making a search for all people named Smith, you may have several Smiths listed. Pressing ENTER continues from the first to the second, and on to the end. After three or four Smiths have been displayed, you can type "R" of the Display Options to RETURN to the display of the first one. A Return to previous files is the same thing as making the search over again from the top. The difference is you don't have to retype the same search command over again. Press "R" and the computer does it for you.

"E" TO EDIT
If you notice that some changes should be made to the displayed file, press "E". This puts the computer in the EDIT mode. The procedure is identical to that of adding new files. The blinking ">" is activated, and you move it up or down using the arrow keys. As in adding files, lines can be altered, added, or deleted. Remember that to delete an entire file, you delete the first line of the file. After you delete an entire file, the program jumps to the Main Add/Search Menu. The jump to the Main Menu also occurs after you press "C" to CLOSE the file.

"N" TO BEGIN A NEW FILE SEARCH
When you are ready to input a new Search Command, or if you want to initiate a special function from the Main Menu, type "N" for NEW. This breaks ZX PRO/FILE from its present search and places the Main Menu before you again.

SPECIAL FUNCTION: AUTO-SEARCH

The AUTO-SEARCH mode does two things. First, it can send out more than just one file to the printer. Second, if you want your files displayed or printed in an ordered fashion, AUTO-SEARCH can arrange the output based on numerical data held in the files so that the lowest file within a specified range will be displayed first. Screen displays or printer output then proceeds with the next larger number. Output continues until the maximum specified number is reached.

The AUTO-SEARCH function is activated by typing "AUTO" at the Main Menu. The computer then asks:

PRINT OUT?

Your answer will be either a "Y" or an "N". A yes answer will result in all found files going to the printer rather than the TV. Next, ZX PRO/FILE asks you to input a Search Command. This can be a single or multi-word command. It should be entered just as when you start a search in the non-auto mode.

If the AUTO-SEARCH function is to provide an ordered display, the computer needs a number at which to begin its search. The next prompt asks you to input this number:

START NO.?
OR ENTER

If you press just ENTER and do not input a number, ZX PRO/FILE assumes that you do not need an ordered display. Instead, the program progresses through a normal single or multi-word search. But.....if you typed "Y" back when the computer asked if you wanted a PRINT OUT, the found files are sent to the printer rather than to the TV screen. Every file in memory that matches with the Search Command will be printed under the parameters set.

If you use ZX PRO/FILE for a mailing list, the AUTO-SEARCH function will produce labels of whatever part of the list you want. Enter a Search Command for "*" to obtain labels for everyone. An AUTO-SEARCH for a state, product, date, etc. will give you mailing labels for all those from a specific state, or those who bought a certain product, or those who ordered on a given date.

ORDERED SEARCHES

If you did type a number when the prompt asked START NO.?, the number you input will be the beginning number of the ordered display. Next, ZX PRO/FILE asks for an END NO. so that it knows what range of numbers it is to look for.

Ordered displays are based on numerical data stored in the last word of your files. You must put a number as the last word of a file if you want to be able to produce ordered displays of the file in question. You do not need to put a number at the end of all files, just the ones you want to be able to order. The example,

THOMAS B. WOODS
P.O. BOX 64
JEFFERSON, NH 03583

has a number (03583) entered as the last word of the file. Therefore, an ordered search for names ordered by zip code could be accomplished. Zip codes, however, are not the best use to which you could put ZX PRO/FILE. The reason is because the range of numbers necessary to accomplish a complete search is so great, the search time becomes excessive. If your mailing list is compiled of names with zip codes scattered from 00000 to 99999, ordered displays would be possible, but if your lowest zip was 27324 the computer would have to make 27,324 searches before it found the first name! Surprisingly, the more names the program holds, the less serious the problem appears because the chances of you having lower zip codes increase. In general, it is best to avoid searches that you know will be fruitless and time consuming.

Some examples of other files that benefit greatly from ordered displays are:

*WRITTEN TEXT
BODY OF TEXT
PAGE 1

*RESEARCH DATA
EXPERIMENT RESULTS
TEST NO. 115

*JOB FILE
PRIORITY 10

RADIO STATION BROADCAST
SCHEDULE. TIME GIVEN ON A
24 HR BASIS: 0245

*SALES RECORD
MONTH 12

*REAL ESTATE FILE
COST (IN THOUSANDS) 78

Any quantity, priority, amount, date, or other item which can be translated into a number can be accessed in an ordered fashion.

PRINTER OUTPUT AND HOW IT RELATES TO "DEFP"

ZX PRO/FILE's printer output works with parallel printers using the Memotech interface. The print format which is defined when you type DEFP from the Main Add/Search Menu lets you specify:

1. The first file line to print
2. The total number of lines to print
3. The number of spaces to skip before printing the next file.

Once set, the print format remains fixed until you change it. The ability to define what and how much of each file to print is a useful feature if you use a dot-matrix printer. You could use ZX PRO/FILE as a customer file in your business. Each file could contain the customer's name and address as well as all other pertinent information. If the file consists of all 12 lines, you could tell the computer to print the entire file for hard copy of your customer records. Then if you want to produce mailing labels, you can change the print format so that only the first four lines of each 12 line file would be printed. Set the SPACE between forms so that the printer jumps to the next blank mailing label. An AUTO-SEARCH for all your customers would then produce the labels, assuming you reserve the first four lines for the customers name and address.

If you use a Timex or Sinclair printer, you can still utilize ZX PRO/FILE's print capabilities. Instructions for modifying the program are given in a later chapter.

SAVING YOUR UPDATED FILES ON CASSETTE

After you actually put ZX PRO/FILE to work, you must always remember that every time you ADD or EDIT files in the program, the cassette becomes obsolete. You should get in the habit of periodically SAVING your updated files.

The Main Menu lets you automatically record a new copy. If you type "SAVE" instead of a Search Command word, the computer SAVES itself. There are no prompts or pauses after you enter the word "SAVE" so make the connections and start the recorder before you hit ENTER.

ZX PRO/FILE uses just about all of your 16K memory. As a result, SAVE and LOAD time is over 6 minutes. Be sure there is enough tape in the cassette. It is probably a good idea to have two or more copies of ZX PRO/FILE. That way, if one is flawed, you'll always have a backup.

If loading time is a problem for you, take heart. ZX PRO/FILE is compatible with many of the "QUICK LOAD" type programs that are available. If the load program you want to use stores its code above RAMTOP, ZX PRO/FILE will probably work with it.

See the chapter on "Program Modifications" for instructions to use FAST LOAD, a top quality utility from International Publishing Software.

MAKING ZX PRO/FILE WORK FOR YOU

Now that you have an idea of how to put information into ZX PRO/FILE and also how to search for and display that information when you want it back again, you need to ask yourself:

A) What exactly do I want to store in ZX PRO/FILE?

B) How can I set up my files so as to make the most of the program's file finding capabilities?

C) How can I file the largest amount of data in the least amount of memory space?

The answers to these questions are different for every individual. ZX PRO/FILE was written for speed, versatility, and capacity, but without at least a little forethought on how to best enter your files, you might miss out on some of ZX PRO/FILE's capabilities. Here are some examples that might give you some ideas.

Customer/Client Record

Suppose you are a sales person and you rely heavily on the telephone for communications with your clients and suppliers. What you need is fast access to telephone numbers. Also, a brief history of past dealings with each customer would be nice. Some of your clients require regular periodic service and you need to know WHEN to call as well as just WHO to call. Finally, if a customer calls you with a problem or request, you need to know where to find all pertinent information about the customer and the situation.

A phone directory of just names and numbers would be an elementary task for ZX PRO/FILE. You could store your names and numbers as numerous one line files which contain both the name and the number. If you could keep each file limited to a maximum of one line, you could store something over 400 names and numbers with the 16K RAM attached.

At any time, it would be a simple matter to make a single word search for the name of the person you wish to call. Type a person's name or a portion of it--every file that the name is found in will be printed one after another.

If your customers buy several products from you, it might be useful to include information on just what they buy along with their name and phone number. You could add more lines to each file. In great detail you could insert the date of purchase, item, amount, and whatever else you need.

Remember, however, that as you add more data, you slowly deplete the memory space available to the computer. There are ways to maximize this limited space by devising codes for certain entries.

Coded Entries Save Memory Space

If you sell 10 different products and you need to record who buys what, and when they buy it, you could enter your clients' phone directory as usual. Then for each client, you could assign a date. With the date could be codes representing your customers' purchases.

ZX PRO/FILE's search mechanism does not distinguish between a coded entry like "C" --meaning "CATALOG REQUEST" or the letter "C" in a file called "CHURCH DIRECTORY". If you input "C" as a search command, every file with a "C" in it--regardless of meaning--will be displayed.

Coded entries (individual letters that are intended to mean something else) could be inverse characters. Thus a search for "inverse C" would eliminate displays of unrelated files.

Another way is to separate code letters with another character (a code separator) like a comma or semi-colon. Then, don't search for just "C". Hunt for "C;" instead. As long as you stick to one method of inputting various letter codes, you can make searches for the code letter with the code separator to print just the files you want.

Here is an example of a "customer file":

```
*P
H.P. COLTER
RD1, BOX 222
JEFFERSON, NH 03583
4-28;C;9.95;DF;
5-15;>;4.95;P;
```

Lines 5-6 of this file each hold 4 pieces of information about Mr. H.P. Colter. Each code is separated from the next by a semi-colon.

First is the date of the transaction
Second, the origin of the order
Third, the amount of payment
Fourth, the item ordered

The breakdown for each line is:

line	code	What it means
1	*	-Beginning of file
1	P	-Product code that gets printed on the mailing label This code means "send this person "PHONE BOOK"
2-4		-Customer name and address
5	4-28;	-Date of transaction
5	C;	-Signifies a response to SYNC Magazine
5	9.95;	-Amount sent
5	DF;	-Product ordered (Data Finder)
6	5-15;	-Date of 2nd transaction
6	>;	-Signifies a response to the catalog
6	4.95;	-Amount sent
6	P;	-Product ordered (Phone Book)

A great deal of information about a business's sales can be gleaned from files storing data in this manner. Mailing labels of any particular group of customers could also be produced.

If your entire customer list were set up in this fashion, you could initiate search commands like:

Command	To tell you
spaceNHspace	all customers from New Hampshire
4-28	all transactions on April 28
C;	all response to ads in SYNC Magazine
>;	all orders from the catalog
DF;/C;	all orders for Data Finder from SYNC (multi-word search)
COLTER	all customers named Colter
NHspace/DF;	all Data Finder customers from New Hampshire
4-	all customers who ordered in April
COLTER/4-	all customers named Colter who ordered in April
*P	all customers whose last shipment was Phone Book
DF;/P;	all customers who ordered both Data Finder and Phone Book
*	all customers filed

Any search can be displayed on the TV screen or sent to the printer. If the print parameters are set so that only the first four file lines are printed, a mailing list can be put on labels. When you actually get down to the business of stuffing envelopes, the first line of each label (the *P in this example) tells you what product to send.

The data analysis possibilities of ZX PRO/FILE would be greatly enhanced if the program could actually count the number of found files and give percentages so that if you entered a multi-word search, the computer would tell you that of all the files checked, it found NN number of files containing the first word. Of those, XX number also had the second word in them. Resulting percentage is ZZ %.

As written, ZX PRO/FILE does not give you this counting capability. Check the "Program Modifications" chapter to find out how to add it if you need file count/tally features.

You should avoid using the "/" in your files because this is one character you cannot search for. This symbol puts the computer in the Multi-word search mode when you put it in a search command word. If you want to store dates in ZX PRO/FILE, separate the month from the year with a dash (-) instead of a slash.

If some of your customers want you to take their orders on a regular day of the week or time of the month, you could insert "WED" or "25TH" somewhere in your customer's file. Every day you could enter the date as a Search Command. ZX PRO/FILE would display the customers that need service for that day.

ZX PRO/FILE the Information Directory

One problem with ZX PRO/FILE and the Timex Sinclair computer is that until really large memory capacity is available, there is simply no convenient way to store huge amounts of information for larger businesses. One way you can use ZX PRO/FILE to work around this limitation is as a catalog file rather than an actual holder of mass megabytes of data. Instead of using the program to hold information, let ZX PRO/FILE tell you where to find it.

Let's say you have a room full of files, records, and general clutter. Records could be stored under any number of categories. You need some way to index all your files so that once you decide what you're looking for, you can figure out where to find it. Now comes ZX PRO/FILE.

Every time you have a piece of literature, a receipt, or a client record that needs filing, log it into the computer. Along with the name and type of information, include the date filed, and what drawer, closet, or shoe box you stuffed it in.

When you need that piece of information back again, you could search through the computer under name, date, or type of information to find the drawer to look in. Similarly, when you're cleaning out your files and you come to box number 47, you might ask "What IS all this stuff?" ZX PRO/FILE could tell you.

ZX PRO/FILE the Product or Collection Catalog

One use similar to the Information Directory is the Collection Catalog. You could store product descriptions for fast easy response to a customer request. Your collection of stamps, coins, or valuables could be similarly logged so that you would know at a glance what you have and what you need.

Using a slide collection of native wild flowers as an example, here is one of many possible ways you could set up your files:

Slide collection sample file

description	file
Botannical name;ht, color;	*VALERIANA OFFICINALIS;4;V;
common names;	VALERIAN;PHU;
special characteristics	SCENTED ROOTS
herbal uses	TRANQUILIZER ANTI-SPASMOTIC PAIN RELIEVER
slide number	NO. 126

Notice how in the first line coded entries are used for height and color. After your collection of slides is logged, you could initiate different searches to select slides meeting your criteria. For example:

A search for:	VALERIANA	would list all slides of a certain genus.
	4;/V;	yields all that grow to four feet AND have violet flowers
	PAIN	selects all species used in the treatment of pain

Also, you could look up plants by their common names or, if your were giving a slide presentation, you could initiate an AUTO-SEARCH to list every slide ordered by slide number:

Type AUTO
NO printer output
issue a search for "*" (all files)
Type "1" for the starting number
Type in the number of slides you have for the end number.

Even if your collection is logged in a different order than the slides are arranged, an ordered search will display them in the proper sequence.

A Product Catalog with Price Stickers

Here's an offshoot of the mailing label technique and the collection catalog. Imagine this scenario. You are an art dealer. You have several hundred oils, watercolors, and screen prints that you sell at art shows. Your collection constantly changes and clients are always asking you to find something you haven't got.

ZX PRO/FILE could neatly arrange your collection and provide you with answers to questions like:

Do you have "BIRDS" by Claude Washington
Or was that "BIRDS" by Clara Washington?
What do you have titled "BIRDS"?
You have "BIRDS" by Clara Washington for $1200?
What do you have by Clara Washington for $500?

For each piece in your art collection you could enter a file like this:

*BIRDS BY CLARA WASHINGTON
OIL $1200

Any of the different word search combinations outlined in the previous sections would provide a wealth of information about what you had or needed.

If you log the price as the last word of the file, you would be ready for the investor who comes to you and asks: What do you have in the $2000 price range?

A simple ordered search between 1950 and 2050 should satisfy your client.

If your collection is going on display and you need labels, ZX PRO/FILE could make them for you. Define the printer status so that the name and price will be printed on a label. Set the space between forms so that the printer will jump to the next label before printing the next file.

If all your files consist of the first two lines only and you are using standard labels measuring 3 1/2 inches by 15/16 inches, Print Format would be set to the following:

first line to copy: 1
number of lines: 2
number of spaces: 4

When the printer format is set, type AUTO from the Main Menu. Type "Y" for "yes, print out". Make a search for all files by entering the "*" as a search command. Finally, when the computer asks for a starting number, press just ENTER. Presto! all your labels are made.

If you're not sure just what pieces you want labels for, you can make them by pressing "C" (for COPY) from the Display Options given after each file display. Labels will be the same, the only difference is that they won't be made automatically. You must press "C" for every label you want.

Remember that the AUTO printout can select certain files with one or two word searches just like when you are displaying the files on the TV screen. You can even print out just those works within a given price range ordered from lowest to highest. If your next show is to feature oils by Clara Washington, the search command that would produce labels is CLARA WASHINGTON/OIL.

ZX PRO/FILE the Bibliography Compiler

The indexing faculties of ZX PRO/FILE can be useful in many tasks. Suppose you're researching a topic for a book you're writing. In the course of your studies you read hundreds of books and periodicals, each with tidbits of information useful to you.

As you find material for your book, you could record the relevant information such as the subject, title, author, date of publication, page number, and where your reference material can be found.

When you need to look up a particular piece, ZX PRO/FILE can tell just where to find it. Enter a subject as a Search Command for a listing of all literature in that field. Enter an author's name to list all works by that author that you have logged. Enter a multi-word search for a subject and author to select just the information you need.

When your book is finished and you're compiling a bibliography or foot notes, all the data you need is available with the push of a few keys of the computer.

Construct an Index

As your book is being written, put items that you think should be included in an index in ZX PRO/FILE's memory. Log the item to be indexed and the page number. When the book is complete, you'll have the index in no particular order.

Although there is no "built-in" alphabetizing function in the program, you can order your files alphabetically using the EDIT function. Here's how:

Whenever you push "E" of the Display Options to EDIT a file, the first thing the computer does is delete the entire file from memory. Then when you CLOSE the file, it gets ADDED back into the first unused space. To alphabetize your data, make a search for "*A".

Push "E" to edit the first displayed "A" file. Then, as soon as the cursor starts blinking, press "C" to CLOSE it. This will put the "A" file in the last file position. Repeat this search/edit/close procedure for all your "A" files. Then make searches for "*B". As you progress through the alphabet, the A's which started at the end are now at the beginning. The "Z's" will be last.

After every file has been alphabetized, initiate a non-ordered AUTO SEARCH for "*". With printer output you'll get hard copy of your book's index.

Research Data and Statistics

ZX PRO/FILE is an excellent tool for storing and analyzing lab or statistical results. Suppose you are conducting an informal study of the effectiveness of certain drugs to combat a specific disease. You could enter daily information about the people being studied: symptoms, medication received, and other relevant data. If you add a new file for each day the data is being taken, ZX PRO/FILE's AUTO SEARCH mode might be a useful way to determine if one medication is more beneficial than another.

If you reserved the last word of each file for the day the data was taken, a numerical search could indicate symptoms displayed by an individual over a period of time. A search such as:

DRUG X/SYMPTOM Y

combined with a numerical search between days 3 and 8 after treatment began would select only those patients who received drug X and displayed symptom Y between days 3 and 8 of the study. By reviewing the number of individuals who fit the categories you specify, certain conclusions might be drawn, or the results may at least point to areas where further study is warranted.

Another approach might be to use the last file word to indicate the "degree of health" on a scale of 1 to 10 with 10 being a complete recovery. In this way a search for:

DRUG X/DAY 6

and a numerical search between 7 and 8 would show those individuals that were well on the road to recovery by the 6th day and received drug X in the course of their treatment.

The option given in "Program Modifications" to count, tally, and give percentages would be particularly useful if you want to use ZX PRO/FILE to analyse research or statistical data. The additional program lines provided will enable you to summarize all data of a particular search.

For single word searches, this option simply counts how many files fit the Search Command. In multi-word searches, the program counts how many files contain the first word; of those, how many also hold the second. Besides the actual numbers, the program gives the percentage:

(n)% of all files holding word A also hold word B

ADVANCED FILE MANIPULATION

After getting this far into the manual, you still may not be sure of the best way for you to input your files, but you probably have a few good ideas. The examples so far demonstrated file building techniques. These that follow show a few ways in which files can be manipulated.

Files Larger Than A Full Screen

The display of files is limited to 12 lines because room on the TV screen must be reserved for printing the Display Options. This does not mean that you can't store files that are longer--only that 12 lines can be displayed at one time.

If you want to keep lists or tables in ZX PRO/FILE which run over the 12 line limit, create two files with the same name. Put the first 12 lines in the first file and the remainder in the second. To access two or more "pages" of such data, simply type as a search command the name which is common to each page. For example:

*BIGFILE

After the program finds and prints the first *BIGFILE it comes to, press ENTER of the Display Options to continue the search. The second, third, fourth, or even more *BIGFILEs will be displayed one after the other.

Since the order in which the files are stored can shift if you should EDIT one, a simple word search may not always result in the files being displayed in the same order. You can insure that page one preceeds page two by entering an ordered search for the page numbers of the files. Just be sure to put a number at the end of each page. Then, enter a one or two word search in the AUTO SEARCH mode. You can tell the computer to print every page beginning with page 1, or you could select pages 5-10 of a large 15 page file.

After you have stepped through the first few pages of a large file, you can easily return to the first again by pressing "R" of the Display Options. This RETURNs you to the first file display without having to re-enter the Search Command again.

Space limitations of the TV prevent the video display of files larger than 12 lines at a time, but if you use a printer in conjunction with the program you can "merge" separate files of the same name into one. Here's how:

A) Put the same file name on the top line of each separate file.
B) Fill all but the last line with your desired information.
C) Set DEFP so that the printer will print out ten lines of every file (lines 2-11) with no space between forms.
D) Use the AUTO-SEARCH with printer output. Issue an ordered search for the file name (line 1) ordered by page number (line 12).

Since the Print Parameters are set so that the first and last lines will not be printed, and there will be no space between each printout, the printer will merge the text of two or more files into one large one. The file names and page numbers will be omitted.

File Adding Tricks

Until now, file ADDING was accomplished by pushing "A" from the Main Menu. There is another way to do it from inside the EDIT mode. Suppose you are looking at one file and you decide to ADD a new one, but you would like to be able to see some of the information in the present display to help you create the new file.

With the present file displayed, press "E" to edit. Drop the cursor down to the first empty line and push ENTER as if you were going to add a new line to the file. At this point you could add a new line, but if you want to ADD a new file, type an asterisk (*) followed by what you want the top line of your new file to be.

Since every file begins with this "*", you can place it "inside" an existing file to separate information into different files.

Using this same technique, you could ADD 12 separate files at the same time just by placing an asterisk in the first column of each line. The computer will treat the lines as though each is a distinct file when you go to look them up again.

This method of adding files would be particularly useful if you use the program to store numerous one line files. It would eliminate the need to type "A" and "C" for every file you want to add.

Another time to use the "*" is in "file splitting". If you have an old file and you have added so much information to it that you feel it should be "split" into two separate files, you can divide the file at any line. Just EDIT the old line so that it begins with a "*".

Combine Different Types of Files in the Same Program

The beauty of ZX PRO/FILE is that it doesn't limit you to one set format or file size. You can hold tiny 2-3 character files right along with mammoth full screen displays without wasting memory space.

You could have a telephone directory, research data, several "pages" of text or tables, and your collection of baseball cards all in one program.

Total capacity is your only restriction. The Main Menu gives you a constant tally of the space remaining measured in "slots". Each slot holds one character. Also, every line uses an additional slot.

Do not try to hold more than space allows. Program failure will result. As you approach capacity, start deleting the old obsolete files. You'll be surprised how the garbage accumulates.

FILE SELECT SUMMARY

From the Main Menu you can:

1. Input a SINGLE WORD SEARCH command.
   This can be any word, symbol, or phrase up to 28 characters long. Every file that contains the search word will be displayed in the order that the computer finds them. A search for "*" will display every file.

2. Input a MULTI-WORD SEARCH command.
   This is the same as a single word command except that the computer looks for two words. Use a "/" between words when you issue this command. Only files that contain BOTH words will be displayed.

3. Initiate an AUTO-SEARCH in either the single or multi-word search modes. Type "AUTO". Prompts follow which ask if you want printer output (a "Y" or "N" will be your answer), what your search command will be, a starting number and an end number.

   If you do not enter a starting number (you press just ENTER instead) the search begins for non-ordered files. In this search, if printer output was requested, all files go to the printer rather than to the TV screen.

   Entering a starting and ending number in AUTO-SEARCH will produce an ordered display based on numerical data which occupies the last word of the files. Files will be printed or displayed only if the file contains a match to the required search command AND a number occupying the last word of the file equals the number the computer is looking for.

   Searching begins with the starting number and progresses through to the specified end number. Search numbers can range from 0 to 99999. They must be positive integers. Only files with a number as the last word can be accessed using the ordered AUTO-SEARCH function. The number stored in the file must be separated from the previous file word with a space.

   Example: PAGE 10 correct format
   PAGE10 incorrect format.

The search methods ZX PRO/FILE uses give you quite a lot of control and selectability in accessing the files you keep. The methods you use to create files, however, have a direct bearing on the usefulness of the program's various search modes.

You should experiment with different file formats and search techniques before you start any serious file construction. You may discover a particularly useful way to store and retrieve your data, and it would be better to make the discovery before half of the files are added rather than after.

HOW THE PROGRAM WORKS
A GUIDE TO THE LISTING

Any program, regardless of how good it is, would be greatly improved if information is added to show you how to change it to meet your individual needs. This section gives an overview of the concepts and techniques used in ZX PRO/FILE. The text will best serve as a guide to the listing rather than a "stand alone" explanation. To get the most from this guide, put the listing in one hand, the text in the other, and watch the program work as you proceed.

ZX PRO/FILE blends Basic and machine language so that all the time consuming tasks are carried out in machine code. Basic generally prints the menus and arranges information the machine language needs to operate.

Understanding this Orchestrate-in-Basic/Process-in Machine Language methodology will make understanding ZX PRO/FILE much easier. The Rube Goldberg type program lines raise flags, insert numbers, and create conditions so that when the jump to lightspeed is made, the target is reached with accuracy.

You'll notice that the text follows a natural flow of the program's operation. Occasionally some program lines will be irreverently skipped by the text. This is because the lines do not perform a significant function in the current process or context. Explaining them would only serve to add confusion to the already difficult task of understanding the program. They will be covered at a later point when the lines actually perform a vital function.

How Data is Stored

After you add a few files to ZX PRO/FILE, stop the computer and type in the immediate mode (no line number):

```
PRINT D$
```

Here is where your files are kept; in a large array--D$(11000). If you scan through the Basic, you'll discover that in only one place is there any reference to D$. That is in line 18:

```
. . ."SPACE OPEN:";LEN D$-P;" SLOTS";. . .
```

This is because D$ is so large, machine code speeds are needed to place and remove characters of the array. Machine code manipulates the contents of D$ when you:

Make a search,
Display a file
ADD or EDIT a file

If Basic were to handle these functions it would take many minutes to complete the operation. To illustrate this time requirement, try the

following short program sometime. It shows just how long it takes for the computer to count to 11,000--never mind process characters of a string in between counts.

```
10 FOR X=1 TO 11000
20 NEXT X
30 PRINT "DONE"
```

The only thing that D$ does in ZX PRO/FILE is reserve space. The memory locations that D$ occupy are always the first 11,000 bytes after the address specified in VARS, the two byte system variable at address 16400 and 16401.

Perhaps you remember reading that "variables move around" in the computer's memory. To use machine code to act on a block of memory that moves around might sound difficult at first, but depending on how you manipulate a variable, you can make machine code programming fairly simple.

When you place a variable into memory, its location does change when you use Basic to alter the variable's value. Also, when you change the length of the program the entire variables section of memory moves up or down to accomodate the new program length.

ZX PRO/FILE's D$ address changes as you add or delete program lines, but since it is never altered by Basic, and it is the first variable entered into the computer, D$'s memory location with respect to the system variable, VARS, is always the same. The first byte of the D$ array is always 6 bytes above the address held in VARS. This bit of knowledge is essential to the computer when it makes a search, adds, or edits a file.

How Files are Added

ZX PRO/FILE has a Basic variable, P, which counts how many spaces in D$ have been used to hold files. It "points" to the last file character. The line referred to earlier:

```
. . .SPACE OPEN:"; LEN D$-P;" SLOTS . . .
```

illustrates how P works. The length of D$ (or 11,000) minus P (or the number of characters presently used in files) indicates how much space remains.

As files are added to D$, a machine language routine takes each file line, transfers it into the first open spaces of the D$ array, and adds to P the number of characters each line has in it. This is done by line 635:

```
LET P=P+USR 16738
```

Exactly how it's done will be described in the EDIT section. Editing and adding functions are almost identical. Both add files to the unused portion of D$. In the case of Editing, the computer deletes the old file first. Then it adds the file back on to the end of D$.

In order for the computer to be able to distinguish one file from the next, or one file line from any other, several important markers are inserted into the files as they go into the D$ array:

The first byte of every file is marked with a "*" to distinguish the end of one file and the beginning of the next.

The last character of every file line is similarly marked with an inverse pound sign (£).

These crucial markers are used extensively in the search and display routines as you will soon find out.

How Files are Found

Lines 17 to 23 display ZX PRO/FILE's Main Menu and let you input a search command into X$. Then, after a few checks are made in lines 25 through 35 for valid entries or special functions, line 36 prints X$ on the clear TV screen.

The cryptic line 40 and the USR call at line 45 work together to place the contents of X$ into a command word buffer, 30 bytes beginning at address 16514 decimal. Line 40:

```
LET X$=X$
```

is a technique that sets the system variable STRLEN (address 16430) equal to the length of X$. If you were to input a 6 character word into X$, STRLEN would equal 6 after line 40 is executed.

If you ever have the urge to study how Basic routines alter various system variables, try this exercise to illustrate how a string affects STRLEN:

```
10 INPUT X$
20 PRINT X$;TAB 10;LEN X$;TAB 20; PEEK 16430+256*PEEK 16431
30 GOTO 10
```

Run the program and input simple strings of varying lengths into X$. The program will print X$, the length of X$ and the value of the system variable STRLEN. After you see what's going on, insert a line:

```
15 LET X$=X$
```

Notice how LET X$=X$ affects STRLEN.

Getting back to ZX PRO/FILE, the following scene is set prior to executing line 45:

STRLEN is set to the length of X$
The address 16883 is poked to zero (by line 19)
The characters of X$ are located in the first few bytes of the display file (remember, it just got printed)

Now with these three bits of information, line 45:

RAND USR 16820

Takes the Display File memory block that contains X$--the length of which is determined by STRLEN--and transfers a copy of it, one character at a time, into the command word buffer. Then it places an inverse period (CHR$ 155) at the end of the word. This character tells the program's search routine it has come to the last character of the command word.

With each character transfer, the routine checks for a "/" (a multi-word search command) and if it finds one, the routine loads the characters that follow into the second word command buffer. This buffer begins 16 bytes after 16514. Then an inverse period is placed at the end of the second word. Finally, the routine loads address 16883 with 155. When this address is anything other than a zero, ZX PRO/FILE knows it is to perform a search for the second command word as well as the first.

Lines 75 and 80 poke the value of P into a 2 byte program variable called BC_CTR. This variable is used by the search routine that follows to determine how many bytes of data need to be checked before the search is complete.

Now the computer is ready to scan the D$ array. To summarize the action so far:

1. The command word is entered into X$
2. X$ is printed and transfered into the command buffer
3. The number of bytes to check is loaded into BC_CTR

Now the linear search routine at line 115:

LET B=USR 16546

zaps through the files. USR 16546 takes the first character of the command buffer and compares it with each character of the D$ array.

If the two don't match, the search through D$ continues. If they do match, the computer matches the second character in the command buffer with the next byte in D$. With each matching character the process repeats until one of two things happens:

A) The inverse period is encountered in the command buffer (indicating a complete match); or

B) The command character fails to match with the data character (indicating a non-match).

If case A holds, ZX PRO/FILE stores the address where the word was found and the number of still unchecked bytes of data. Then the computer steps backwards through the D$ array until it finds an asterisk (the beginning of the found file). The address the asterisk is found in is stored in the unused two byte system variable at 16507. For clarity, let us call this variable FILE PEEK.

Every time USR 16546 finds a word in D$ that matches the search command word, FILE PEEK is loaded with the starting address of the found file.

If case B holds--the search results in a non-match before the inverse period is encountered--the routine simply continues scanning files for another matching first character in the command buffer.

Searching continues until the last byte of data has been checked. Then the address of the first file is loaded into FILE PEEK. The first file is one line of characters. It says:

```
* SEARCH IS COMPLETE
```

In this way, a search always results in a file's address being loaded into FILE PEEK. It will either be a file in which a match is found, or it will be the first file of D$, which indicates that a complete scan of the data was performed. As files are checked, the BC_CTR counts down the number of unchecked data bytes and this number is what B equals when line 115 says LET B=USR 16546.

In every case but one B will be less than the total value for P because B starts off with P's value and counts backwards. But if the command buffer holds:

```
WOODS
```

and the last few characters of D$ are WOOD, the search would produce matching characters, decrementing the BC_CTR right past zero. The registers inside the CPU count in binary and the number they record after decrementing zero is 1111111111111111. Definately greater than P.

Thus, even though every byte of data has been checked, the unthinking computer thinks it has more data to search since its BC register pair doesn't have a zero in it. The routine continues searching other bytes of extraneous data, usually until it finds WOODS in the variables section of memory where WOODS is held in X$. The result is a fooled computer. It thinks it found a file and if were printed, a lot of garbage would be shown on the TV screen. Line 117 prevents this from happening. A "garbage file" always results in B>P so line 117:

```
IF B>P THEN LET B=USR 16602
```

sends the computer off to the part of the search routine that places the "SEARCH IS COMPLETE" file in FILE PEEK. This routine also sets B equal to 1.

MULTI-WORD SEARCHES

If the search command buffer loader routine (USR 16820) puts a 155 in address 16883, a search for a second word in the found file is needed.

Line 121:

```
IF PEEK 16883 AND B<>1 THEN IF USR 16780=0
THEN GOTO 115
```

searches for that second word. Address 16883 will hold either a zero or 155. Line 121 first checks this address. If it is not zero and if B is not 1 (search is not complete), USR 16780 is called upon to search the file listed in FILE PEEK for the second search word. If the routine finds it, USR 16780 will equal 1. If the second word is not found the USR routine will equal zero. Therefore, non-matches go to line 115: continue searching more files for the first word. Matches pass to line 125 for printing found files.

FILE DISPLAY

For found single or multi-word file searches, the only thing left to do is print the file. Lines 125 to 160 do just that.

First, the search command is printed: X$. Then, line 130 calls a machine code print routine (16688). This routine steps through the found file beginning at the address held in FILE PEEK. It looks for either an asterisk or an inverse pound sign and if it finds them the routine returns to Basic, X being the address of the current file character.

Every character that is not a "*" or a "Ł" is placed one at a time into the accumulator of the CPU. The RST 10 command is applied. RST 10 (Restart 10 in English) is a handy ROM subroutine that takes whatever character is in the Accumulator and prints it.

The next Basic lines, 132 and 140, check to see what caused the return to Basic. If it was the "Ł" (end of line), line 132 simply drops the print position down to the next line. When a "*" causes the return, file display is complete because the asterisk is acutally the beginning of the next file. Program line 220 is then executed: Print the Display Options.

AUTO SEARCHES

The fundamental search mechanism in an AUTO-SEARCH is the same as in a regular search, but the program does a little bit more.

Breaking into the AUTO-SEARCH mode is carried out by line 28. When you type "AUTO" instead of a search command word from the Main Menu, you go to line 665.

Here, the computer first determines if you want output to go to the printer. A "Y" answer causes the variable C to become one; a result of line 675. Any other answer will make C=0.

Next, the program asks for a Search Command word. Everything here is the same as at line 23. The command word goes into X$ and it can be a single or multi-word command.

You may be wondering what Q$ is since line 700 (and several other lines throughout the program) prints it 3 times. Q$ is a string of 32 blank spaces. It is used to erase lines of the TV screen much like CLS does only Q$ erases just one line at a time.

Anyway, after the command word is input, line 703 asks:

START NO.?
OR ENTER

The AUTO-SEARCH can do 3 things:

1. Output all found files to the printer as it finds them;
2. Send files to the printer on the basis of numerical data stored in the last word of the files;
3. Display ordered files on the TV.

Let's look at what happens when you type just ENTER first. Line 706 tells ZX PRO/FILE to go to line 27 if you push ENTER. Line 27 resets Z$ to "00000". If you type just ENTER when line 703 asks for the starting number, Z$ becomes 5 blank spaces. It is necessary to put the zeros back because now, from 27 on, the computer will begin searching files for X$ matches. Line 124:

```
IF Y AND VAL Z$>E THEN GOTO 1070
```

would stop the computer dead in its tracks if Z$ did not have a number in it. But we are getting ahead of ourselves. If just ENTER is pressed when line 703 asks for a starting number, the computer goes to line 27 to search.

Since we've already covered this section, we won't do it again. Everything is the same:

The command is loaded into its buffer
The search is carried out
The file is printed

BUT . . . If you typed "Y" for "yes, send output to the printer", the variable C equals 1. When C=1, line 160 sends the computer to line 3000. Since the computer is in the FAST mode, and Display Options are not printed, you never see the file on the TV, but it is there. Instead, 3000 to 3040 Copy the found file as per the printer parameters.

How the printer is activated will be covered later. For now we'll just say that the file gets printed out and once done, line 3030 makes Y$ empty. Next, you go to line 245:

```
GOTO 220*(Y$<>"N")-110*(Y$="")-190*(Y$="R")
```

huh?

With Y$ empty, line 245 says:

GOTO 220*(1)-110*(1)-190*(0)

The numbers in parentheses are the true/false values for the expressions (Y$<>"N"),(Y$=""), and (Y$="R") respectively. If the expression is true, its value is 1; false equals 0.

When you work out the arithmetic, you get:

220-110-0 or GOTO 110

The search continues just as if you had pushed ENTER of the Display Option Menu. Every found file gets copied until the SEARCH IS COMPLETE file is reached. When this happens, B equals 1 so line 160 no longer directs the computer to the copy routine. Instead, it goes right past 160 to the Display Options.

That is what happens when you push ENTER at line 703--the AUTO SEARCH mode. If you type a number rather than just ENTER, you GOSUB 1202, a result of line 707

This is a subroutine that checks each digit you typed to make sure it is a number rather than a letter. It is a means of rejecting invalid entries that could cause the program to stop with an error should your finger miss hitting a number key. What GOSUB 1202 does is this:

Y is set to 1

Then a simple FOR. . .NEXT. . . loop checks each character of Z$ (the number you input). If the character's code is less than 28 (a zero) or greater than 37 (a nine) then Y is changed to a two.

Line 1225 jumps you to 1200 if this happens to be the case. At 1200 you must re-input Z$. You must type numbers or the subroutine will spit it back at you and force you to repeat the input until you get it right.

When Z$ is finally given the OK (Y=1), you return to line 708 where the variables E and L assume the value of Z$. Then line 710 asks you to input an end number. After you GOSUB 1200 to input this valid number, you go to 30 to search. Now:

Y=1
X$= the search command
E and L= the starting number
Z$= the end number

Line 33 becomes functional now because Y=1 but E and L are not changed. They are already equal. This will change as the search progresses and line 33's necessity will become apparent.

After a search results in a found file but before the file is displayed, line 123 causes a jump to line 1000. Here Y$ becomes 5 blank spaces and line 1010 performs its antics to set STRLEN to 5.

USR 16890, initiated by line 1030, actually hunts down Y$'s location in memory and changes it so that it equals the last five characters of the found file as given in FILE PEEK.

Just for fun, insert lines:

1020 STOP and
1031 STOP

Run an ordered AUTO-SEARCH. When the program stops with the report D/1020, type:

PRINT Y$ result: all spaces

Press CONT to run the USR routine and when the computer stops at line 1035, type again:

PRINT Y$

Now Y$ equals the last five characters of the file beginning at the address specified in FILE PEEK. That isn't quite true. If the last word in the file is less than 5 characters in length, Y$ will hold only as many characters as the word does. For accuracy, Y$ equals the last word of the file OR the last five characters, whichever is less.

Lines 1032 and 1033 work together to delete the first characters from Y$ if they should be zeros or have codes of less than 29. Then line 1040 compares Y$ with the variable E. If the characters are the same then GOTO 125 to display the file. If they're not identical, line 1060 sends the computer back to 115 where the search continues.

After the file containing the appropriate number is displayed (or printed if C=1), the search continues for more files that have the same number. All will be displayed.

As long as there is more data to search, line 123 jumps to compare the last file word with E, but when the search is complete (B=1), line 124 jumps to 1070.

The lines 1070 and 1100 increase E by one and the search for the new number E begins.

This whole process continues automatically if files are sent to the printer. If printer output is not requested, the display options are printed and the files are shown on the TV screen one at a time.

If you type "R" of the Display Options--"Return to previous displays"--line 245 says:

GOTO 220*(1)-110*(0)-190*(1) or GOTO 30

Here is where line 33 affects the program. Remember when the AUTO SEARCH started, E equaled L but after several repetitions E was incremented; L stayed the same. Therefore, line 33 resets E back to its original value when you type "R" to return.

Another thing that happens when you punch RETURN--and this happens in the non-AUTO mode as well--is that before the search begins, lines 75 and 80 repoke the BC_CTR with the value for P. The effect is the same as if you were starting a search from the Main Menu rather than continuing from the Display Options. Throughout this rather complex operation of:

SEARCH/FIND/PRINT/CONTINUE

X$, the search command word, remains unchanged. When you hit "R" of the Display Options, the command word is the same as it was. The search simply starts again from the top and previous displays of the same command are repeated.

EDITING

One display option not yet covered is that of editing the file shown on the screen. If you press "E", you go to the EDIT routine at line 300.

Here a machine language call automatically deletes the file from the D$ array and subtracts from P the number of characters the file occupied.

The machine language deletion works like this:

FILE PEEK holds the starting address of the file.

The machine code starts at this address and steps ahead, counting bytes until it comes to an asterisk or the beginning of the next file. Then the code counts through all occupied bytes of D$ until it comes to the last character. With these three pieces of information:

FILE PEEK address,
Address of the next file,
The number of bytes to the end,

the code moves every character from the next file on, up the number of bytes the old file occupied. This operation is easier to visualize than it is to explain.

Imagine that the D$ array is a file card holder with 11,000 cards in it. Each card is numbered and each has one character of a file printed on it. Some cards have letters, others have asterisks indicating a new file, still others have inverse pound's (£) on them to tell the print routine to display the characters that follow on a lower line.

At this point in the program there is some file displayed on the screen. The beginning of the file (*) is the card whose number is held in FILE PEEK. The machine code starts from this card number and checks every

one that follows for a "*" (the next file). When it finds one, the code counts the number of remaining cards. Finally, the computer goes back to the card whose number is in FILE PEEK and replaces the character printed on it with the starting character of the next file. Then the computer takes the next card and replaces the character on it. Every card from FILE PEEK to the end is thus overwritten and the number of cards that were used for the deleted file is subtracted from P--the total number of cards used.

You are entitled to wonder why the file is deleted just because you want to EDIT it. Perhaps you want to lengthen the file.

> Just because the file is deleted from the D$ array, it isn't gone for good. The file still exists on the TV. As long as you don't clear the screen, you still have it stored in the display file.

So after the file is erased from D$, line 320 jumps to line 504. From here on, the process of editing and adding files is identical. The same lines are used for both.

The Basic that runs from lines 504 to 565 manipulates the file that's displayed on the TV screen. Line 504 sets the variable L to the memory address of D$'s first byte (6 bytes after the address specified in VARS). The variable Y represents the line on which the movable cursor blinks. Since the top line of a file is always printed on line 3 of the TV, line 505 starts Y off with the value of 3. This value will change as you edit different file lines. It will range from 3 to 14 representing lines 1-12 of a file display.

When the computer shows moving characters, the SLOW mode must be used. Line 508 puts the computer in SLOW while you're working on a file.

The next 5 lines (509 to 550) cause the cursor to blink. First 509 prints at line Y, column 0; a "greater than" sign (>). Then the same line immediately overwrites it with a space. The effect is one "blink".

Keyboard scanning takes place at line 510. If you push "C" when the computer is looking, line 520 jumps you to 600 to close the file.

The next line (545) changes the value of Y if you press the "6" or "7" keys. Here's how:

> The line checks for two possible key presses but the second half of the line is executed only if Y is within the correct range of values.

If you press the "6" but Y must be less than 14, OR if you press "7" and Y is greater than 3, the expression:

```
LET Y=Y+SGN (6.5-VAL Y$)
```

is executed.

This expression will either add one or subtract one from the value of Y. "Signum", or SGN is a function on the "F" key. It's result is the sign (+,-, or 0) of a number. For a positive number, SGN will be 1; for negatives, SGN is -1; and for zero, it is also zero.

Since line 545 is executed only if Y$="6" or Y$="7", this expression will read:

```
LET Y=Y+SGN (6.5-6)   if you press the "6" key or
LET Y=Y+SGN (6.5-7)   if you press the "7" key.
```

When your work out the math, SGN (6.5-6)=SGN (.5), a positive number if you press "6" (the key with the down arrow on it). So line 545 really says:

If you press the down arrow, let Y=Y+1

When you hit the "7" key (the up arrow), SGN (6.5-7) is the same thing as SGN (-.5), a negative number. In this case:

If you press the up arrow, let Y=Y-1

Line 550 will send you back to 509 if you're not pressing ENTER. Here you "blink" the cursor again at Y's new value. This sequence repeats until you press ENTER. Then lines 552 and 565 let you change the data on the TV screen.

First, you are asked to input a line of text at 555. If Y is 3 (the cursor is pointing at the first file line) and you press just ENTER, you delete the file. Line 556 goes to 650 where an asterisk and a graphic character (CHR$ 134) are placed at the end of the occupied bytes of D$. Remember, when you ADD or EDIT, the variable L is the address of the first byte of D$ (it's set by 504). Therefore, when 650 pokes:

```
L+P,23
```

and 655 pokes:

```
L+P+1, 134
```

the addresses you poke are the end bytes of occupied data in D$. These characters help the program distinguish the end of used data. Since your aim is to delete an entire file when you push just ENTER with the cursor at the first line, there isn't anything else the computer needs to do. Therefore, line 662 jumps back to display the Main Menu.

If you want to change a file line, 560 overwrites the old line of text with the new. Earlier it was mentioned that Q$ is a string of 32 blank spaces, Y is the line at which the cursor is blinking. Thus, line 560 first erases everything that used to exist on line Y, then the same program line prints the new material.

Line 562 automatically advances the cursor to the next line if Y and the length of Y$ are within proper limits. If you input a line of text into Y$ that is too long, or if the cursor is already at the bottom of the screen, Y will remain unchanged.

The expressions inside the parentheses in line 562 have numeric values of one if true, zero if false. So the line says:

```
LET Y=Y+(1)-(0)    if Y<14 or LEN Y$<29
LET Y=Y+(1)-(1)    if Y<14 and LEN Y$>=29
LET Y=Y+(0)-(0)    if Y>=14 and LEN Y$<29
```

Normally, Y will become Y+1 and the program jumps back to 506 to repeat the "blink"/line edit sequence. This repetition continues until you decide to close the file. Then 520 is executed: GOTO 600.

Here, the computer is placed in FAST and a FOR. . . NEXT. . . LOOP adds each line displayed on the TV to the D$ array. For each iteration of the loop:

> Line 610 moves the print position to line Y, column 1. This action sets the system variable DF_CC to the address in the display file which forms the beginning of the line of text.

Line 615 calls USR 16617 that checks to see if the first character of the line is blank. If it is, USR 16617 equals 0 and without further ado, the loop is advanced to the next line. This is the reason why file lines that begin with a space will be treated as though they are completely empty.

Lines 620 and 630 poke the address of the first unused byte of the D$ array into a variable called P POINTER (address 16736 and 16737). Then line 635 adds the line to the array. It also advances P by the correct amount and places an inverse "Ł" at the end of the line. This character tells the computer to drop down one line when displaying found files. You can actually see how this routine works by deleting line 602 from the program. Add a file in the normal way and watch what happens when you close the file.

After 645 and 655 adjust P and mark the end of "used characters", line 662 takes you back to the Main Menu.

PRINTER PARAMETERS

When you type "DEFP" from the Main Menu, the DEFP routine at line 2000 is activated. Here 3 variables are set:

> C1 is the first line to copy
> C2 is the total number of lines to copy
> S is the number of blank line feeds between forms.

The DEFP function makes heavy use of the valid number subroutine beginning at line 1200. After a valid number is input, lines 2025 and 2047 further check the number to make sure you don't try to print more than a screenful of a found file.

These lines prevent you from trying to print 20 lines of data beginning at line 16, for example. Since only 12 lines are allowed per file display, this is clearly an unreasonable request. Numbers input that are beyond the proper range are rejected and you must re-enter them. Once printer parameters are properly set, line 2100 jumps back to the Main Menu.

OUTPUT TO THE PRINTER

The jump to printer output occurs in two places: from the Display Options, line 231 sends the computer to line 3000 if you press "C" for COPY. In the AUTO-SEARCH mode, line 160 jumps to 3000 as well. With the print parameters defined as:

C1=starting line
C2=number of lines
S=number of spaces

lines 3000 to 3040 work in the following manner:

First, line 3000, PRINT AT C1,0; sets system variable DF_CC to the address in the display file where printing is to begin. Then line 3010 pokes address 16679 with the number of lines to copy.

A short machine code routine called by line 3020 takes this information and then jumps into the middle of the ROM Copy routine. Normally the ROM routine sets printing at the top line of the display file and continues through every line. ZX PRO/FILE initializes certain registers of the CPU and then jumps into the ROM with new starting point and number of line to print data.

Lines 3025 and 3030 change the contents of Y$ if the computer is in the AUTO-SEARCH mode to either a "C" or an empty string if B>1. Then lines 3035 and 3037 implement a FOR. . .NEXT. . . LOOP to force the correct number of line feeds before the next file is printed. With printing now complete, the computer goes to line 245.

Depending on Y$'s contents, this line either jumps to 220 in the case of Y$="C", or 110 for more searching if Y$="".

NOTE: If you use a Sinclair or Timex printer, you will have to alter the machine language copy routine. Instructions for doing this are given in the next chapter. ZX PRO/FILE's printer output code is written specifically for the Memotech Centronics I/F which "cuts-in" on the Sinclair 8K ROM program with its own instructions.

PROGRAM MODIFICATIONS

Some users will undoubtedly say to themselves, "ZX PRO/FILE is really great, but what if it could. . . ."

This chapter is about some of those "what if's".

Before any enhancement can be made to ZX PRO/FILE, be forewarned that the way the program is written, every last byte of the 16K memory is used for program lines and data capacity. If you want to tailor ZX PRO/FILE to meet your own needs, you will almost certainly have to either decrease data capacity or obtain a larger memory pack. The function/data capacity trade-off decision is left to you.

If you have files stored in the program, you can't decrease capacity without losing your data so whatever modifications you make should be made before you start any heavy duty file construction. Here is how to begin the modification process.

1. Load the ZX PRO/FILE cassette
2. Exit the program from the Main Menu. To do this simultaneously press the SHIFT and EDIT keys to delete the quotes at the bottom of the screen. Then press STOP. Machine code is located in Rem lines below line 17. DO NOT LIST any line less than 17.
3. Type CLEAR and Enter to clear all variables.
4. SAVE a copy of the cleared program.
5. Add your modifications.
6. Re-enter the variables in the following order. This is done in the immediate mode (no line numbers).

```
DIM D$(with a new number less than 11000) Experiment.
LET P=0
LET Q$=D$( TO 32)
LET Y$=""
LET S=1
LET E=0
LET L=0
LET C1=3
LET C2=11
```

7. Type GOTO 17 to start the program
8. ADD the first file. It should read:
   "inverse space-SEARCH IS COMPLETE"

In order for ZX PRO/FILE to work, the D$ array must be the first variable entered. The order of the remaining variables does not matter. Estimating the number to use in dimensioning the D$ array will be the hardest part of the modification process. It must be small enough to allow for extra programming without the computer running out of memory. Experimentation seems to be the only way to find the right size. To test for proper size, add search and delete lines of files. Input long strings for search commands. If you don't get a report 4 error halt, the array is probably ok.

Make ZX PRO/FILE a FASTLOADer

FASTLOAD, a program available from International Publishing & Software, can dramatically speed the SAVE and LOAD time. Purchasing the program is worth serious consideration. To adapt ZX PRO/FILE, follow these instructions and read the manual that comes with FASTLOAD.

1. Load the FASTLOAD Monitor program. Break, and load the cleared copy of ZX PRO/FILE.

2. Once loaded, change line 25 to:

```
25 IF X$="SAVE" THEN GOTO 4050
```

3. Add these new program lines:

```
4050 DIM T$(24)
4060 LET T$="-"
4070 CLS
4080 PRINT USR 32685
4090 LET T$="inverse P space ZX"
4100 PRINT USR 32685
4110 GOTO 17
```

4. In the immediate mode, type:

```
DIM D$(9700)
```

5. Then enter the remaining variables.

6. Type GOTO 17 and add the first file: " SEARCH IS COMPLETE" all characters should be inverse.

Now when you type SAVE from the Main Menu, the program FAST SAVES. Time: 73 seconds.

Excellent instructions accompany FASTLOAD that show how to reload the program as well as Load/Save just the files. This cuts loading time even more. At the expense of a little over 1K of file capacity, FASTLOAD is an economical way to speed up the loading of ZX PRO/FILE.

File Search/File Count

This adaptation counts files found by a search. Using it can tell you how many files contain a specific word. Furthermore, if you use a multi-word search, this program enhancement tells you how many files contained the first word and of them, the number that also contained the second.

This addition could, for example, tell you how many people in your customer file came back to order a second product after ordering the first. The numbers and percentages are printed with the Display Options. They reflect the tally up to the latest found file. When the SEARCH IS COMPLETE file

is displayed, the totals for all the files are given.

If you want just the final numbers and are not particularly interested in individual file contents, you can obtain this information by initiating an AUTO-SEARCH. Type "Y" when the computers asks, "PRINT OUT?" If your printer is turned off or if you do not have one connected, the search will be completed but will not be displayed or printed out. The result is a display of just the numbers relating to found files. This tabulation of data will be very useful for analysis of your files. To add file counting capabilities, load the cleared copy of ZX PRO/FILE and insert these program lines:

```
 30 DIM G(2)
120 IF B<>1 THEN LET G(1)=G(1)+1
122 IF PEEK 16883 AND B<>1 THEN LET G(2)=G(2)+1
190 IF Y THEN GOTO 220
200 IF PEEK 16883 THEN PRINT AT 15,0;G(2);" OF";
210 PRINT AT 15,8;G(1);" FILES";
215 IF PEEK 16883 AND G(1) THEN PRINT TAB 18;
    100*G(2)/G(1);" PERCENT"
```

After the new lines are entered, re-enter the variables starting with the D$ array. With 16K memories, D$ must be decreased to D$(10500).

ZX PRO/FILE in Big Memories

This program is perfectly compatible with computers using more than 16K of RAM. Certain memory allocations set by the ROM limit data capacity to approximately 45K even if you use a 64K memory pack.

Add these program lines to the end of the cleared copy of ZX PRO/FILE:

```
4000 DIM D$(10505,4)
4010 LET L=PEEK 16400+256*PEEK 16401
4020 POKE L+1,41
4030 POKE L+2, 164
4040 POKE L+3,1
4050 POKE L+4,38
4060 POKE L+5,164
4070 POKE L+6,0
4080 LET P=0
4090 LET S=1
4100 LET C1=3
4110 LET C2=11
4120 LET E=0
4130 LET Y$=""
4140 LET Q$=D$( TO 32)
```

Type RUN 4000, and when the computer stops with report 0/4140 type GOTO 17 to start the program up. Don't forget to add the first file as shown in step 8 on page 40. These program lines dimension an array, 10505 strings-each with 4 characters. The variable L is the starting address of the variables section of memory. The pokes to these first few bytes (4020-4070) change the array so it has just one dimension and 42,022 characters! This method will work where DIM D$(42022) fails.

The routine programmed into lines 4000 to 4100 is a technique that fools the computer into thinking that D$(10505,4) is D$(42022) instead. You can use the same procedure to create strings of any size up to a limit of about 45000 characters. Here's how:

1. CLEAR the computer.
2. DIM a four character string array of the size you want. Ex: DIM D$(10505,4)
3. Multiply the size (10505) by the number of characters (4) and add five. (answer: 10505*4+5=42025) Write down your answer.
4. Initialize a variable (L) to equal the address specified in the system variable VARS (see line 4010)
5. POKE the two bytes after L (L+1 and L+2) with your answer in step 3. To do this type:

   POKE L+1,42025-256*INT (42025/256)
   POKE L+2,INT (42025/256)

6. The next byte (L+3) is poked with the value of one to represent a single dimension array.
7. Finally, you must poke the remaining two bytes (L+4 and L+5) with your answer in step 3 minus three. Ex:

   POKE L+4, 42022-256*INT (42022/256)
   POKE L+5, INT (42022/256)

8. The sixth byte after the address specified in L becomes the first character of your new single dimension array of 42022 characters.

PRINTER OUTPUT WITH TIMEX PRINTERS

Mentioned earlier was the fact that ZX PRO/FILE is written to provide printer output when used with a Memotech centronics interface. When Memotech wrote the ROM software that converts the Sinclair characters into ASCII, they "cut-in" on the Sinclair 8K ROM with their own instructions. The result is ZX PRO/FILE won't work with Timex printers unless you make this minor change. Follow step 2 on page 40 to exit from the program at the Main Menu. Then type:

POKE 16677,0 and ENTER
POKE 16681,118 and ENTER

Type GOTO 17 to start the program running again.

MAKE ZX PRO/FILE DO MORE THAN JUST PRINT FILES

If you have the good fortune to own a larger memory, you can make files actually perform preprogrammed tasks. Such tasks might include a program jump to an arithmetic calculator or to the printing of various form letters. You could even program the saving or loading of new data from a disk. Total data capacity may need to be reduced depending on the extent of your additional programming, but such a function would nevertheless add a tremendous amount of flexibility.

These program additions will cause a jump to lines after 5000 if the "do" file and the Search Command used to find it fit certain conditions:

```
116 LET Q=(1+(PEEK 16507+256*PEEK 16508))
118 IF PEEK Q=13 THEN IF X$=CHR$ PEEK (Q)+
    CHR$ PEEK (Q+1)+CHR$ PEEK (Q+2)+CHR$
    PEEK (Q+3) THEN GOTO 5000+VAL CHR$
    PEEK (Q+4)
```

A "$" character (CHR$ 13) is used to signal a "do file", but it will be activated only if the search command and the first 4 characters of the file match.

For example, you could create a file that looks like:

```
*$FIL1
```

Program line 118 looks for a "$" and a match of "FIL" with X$, the search command. If these conditions are met, the line jumps to line 5000 plus the number (1) which follows. At line 5001 you could begin your new program function or place a GOTO which branches to it.

Using this method gives you 10 different possible "do files". Jumps to 5000 through 5009 are possible.

Lines 116 and 118 are not fool proof. Files must be entered in a specific way or they won't work:

First, a "$" should immediately follow the asterisk.
The next three characters represent the name of the "do file".
Finally, a number must be present in the final slot or the program will stop with an error. This number, added to 5000, will be the line number to which the program jumps.

Using the example *$FIL1, when you input $FIL as a search command, the program line 118 causes the computer to go to line 5001. Here, could be your instructions enabling a disk or stringy-floppy.

APPENDIX I ZX PRO/FILE Basic Listing

(c) 1983 Thomas B. Woods
all rights reserved

```
  17 CLS
  18 PRINT "zx pro\file",,,,"ent
er a search command","or type  a
- to add    ","       save- to sa
ve   ","SPACE OPEN: ";LEN D$-P;"
 SLOTS";AT 10,0;"SEPARATE MULTI-
WORD","COMMANDS WITH A ""/""",,,
"TYPE ""AUTO"" FOR AUTO-SEARCH"
  19 POKE 16883,0
  20 LET Y=0
  21 PRINT TAB 5;"""DEFP"" FOR P
RINT FORMAT";AT 19,0;" PRINT STA
TUS",,"START/NO./SPACE";TAB 2;C1
-2;TAB 6;C2;TAB 12;S
  22 LET C=Y
  23 INPUT X$
  25 IF X$="SAVE" THEN SAVE "Zx"
  26 IF X$="DEFP" THEN GOTO 2000
  27 LET Z$="00000"
  28 IF X$="AUTO" THEN GOTO 665
  31 CLS
  32 IF X$="A" THEN GOTO 500
  33~IF Y THEN LET E=L
  34 IF X$="" THEN GOTO 18
  35 IF X$(LEN X$)="/" OR X$="SA
VE" OR LEN X$>28 THEN GOTO 18
  36 PRINT X$
  40 LET X$=X$
  45 RAND USR 16820
  75 POKE 16600,P-256*INT (P/256
)
  80 POKE 16601,INT (P/256)
 110 CLS
 115 LET B=USR 16546
 117 IF B>P THEN LET B=USR 16602
 121 IF PEEK 16883 AND B<>1 THEN
 IF USR 16780=0 THEN GOTO 115
 123 IF B<>1 THEN IF Y THEN GOTO
 1000
 124 IF Y AND VAL Z$>E THEN GOTO
 1070
 125 PRINT X$;TAB 0;"file search
",,,,TAB 1;"*";
 130 LET X=USR 16688
 132 IF PEEK X=140 THEN PRINT TA
B 1;
 140 IF PEEK X<>23 THEN GOTO 130
 160 IF B<>1 AND C=1 THEN GOTO 3
000
 220 PRINT AT 16,0;"hit enter to
 continue searching  c- to copy
                     r- to retur
n to previous files  e- to edit
this file            n- to begin
 a new file search  "
```

NOTES

Here begins the Main Menu
Note: lower case letters signify INVERSE

D$=the data array

17 CLS gives 0 10 2 0 25 1 118
0 0A 2 0 19 1 76
when peeked 0 11 2 0 245
0 0B 2 0 F5

Input Search Command into X$

Printer parameters set at line 2000

AUTOSEARCH's start at line 665

ADD files at line 500

Load the search command into buffer
Poke BC_CTR with number of bytes to check

Search for match to search command word

Search for second word

AUTOSEARCH's go to line 1000

Display found files

Copy file into display file

When C=1 the printer is activated at 3000

Print display options

```
 230 INPUT Y$
 231 IF Y$="C" THEN GOTO 3000
 232 IF Y$="E" AND B<>1 AND B<=P
-20 THEN GOTO 300
 245 GOTO 220*(Y$<>"N")-110*(Y$=
"")-190*(Y$="R")
 300 LET P=P-USR 16624
 310 PRINT AT 0,20;"edit\file"
 320 GOTO 504
 501 PRINT "add\file]]enter titl
e";AT 3,1;"*"
 504 LET L=6+(PEEK 16400+256*PEE
K 16401)
 505 LET Y=3
 506 PRINT AT 16,0;"PRESS ""C""
TO CLOSE THE FILE.     ARROWS MOV
E THE "">""                HIT ente
r TO INPUT DATA";Q$;Q$;Q$
 507 IF LEN Y$>=29 THEN PRINT AT
 20,0;"data too long]]re\input"
 508 SLOW
 509 PRINT AT Y,0;">";AT Y,0;" "
;
 510~LET Y$=INKEY$
 520 IF Y$="C" THEN GOTO 600
 545 IF (Y$="6" AND Y<14) OR (Y$
="7" AND Y>3) THEN LET Y=Y+SGN (
6.5-VAL Y$)
 550 IF INKEY$<>CHR$ 118 THEN GO
TO 509
 552 PRINT AT 16,0;" INPUT DATA.
 PRESS JUST enter          TO DELE
TE";Q$;Q$;Q$
 553 PRINT AT Y,0;"~";
 555 INPUT Y$
 556 IF Y=3 AND Y$="" THEN GOTO
650
 557 IF Y=3 THEN LET Y$="*"+Y$
 560 IF LEN Y$<29 THEN PRINT AT
Y,0;Q$;AT Y,1;Y$
 562 LET Y=Y+(Y<14)-(LEN Y$>=29)
 565 GOTO 506
 602 FAST
 605 FOR Y=3 TO 14
 610 PRINT AT Y,1;
 615 IF PEEK USR 16617=0 THEN GO
TO 640
 620 POKE 16736,(L+P)-256*INT ((
L+P)/256)
 630 POKE 16737,INT ((L+P)/256)
 635 LET P=P+USR 16738
 640 NEXT Y
 645 LET P=P-1
 650 POKE L+P,23
 655 POKE L+P+1,134
```

NOTES

Make your selection
"COPY" at line 3000
"EDIT" at line 300

EDIT routine starts here

ADD routine starts here

L=first byte of D$ data

Y=line at which the ">" blinks

This line blinks the cursor

You go to 600 when you CLOSE a file

If you don't press ENTER you go back to 509

Otherwise you input data
If Y=3 and you press just ENTER, you delete the entire file

After printing the new file, line Y is automatically incremented

This loop loads the display file into the D$ array.

P is advanced

```
 660 FAST
 662 GOTO 17
 665 PRINT AT 13,0;" PRINT OUT?"
;Q$;Q$
 670 INPUT X$
 675 IF X$="Y" THEN LET C=1
 700 PRINT AT 13,0;"input comman
d";Q$;Q$;Q$
 701 INPUT X$
 703 PRINT AT 15,0;X$;Q$;AT 16,0
;"start no.?";TAB 0;" OR ENTER"
 704 INPUT Z$( TO 5)
 706 IF Z$="      " THEN GOTO 27
 707 GOSUB 1202
 708 LET E=VAL Z$
 709 LET L=E
 710 PRINT AT 17,0;L;Q$;AT 16,18
;"end no.?"
 717 GOSUB 1200
 725 GOTO 30
1008 LET Y$="      "
1010 LET Y$=Y$
1030 RAND USR 16890
1032 IF CODE Y$<29 THEN LET Y$=Y
$(2 TO )
1033 IF CODE Y$<29 THEN GOTO 103
2
1040 IF Y$=((STR$ E)+"     ")( TO
 LEN Y$) THEN GOTO 125
1060 GOTO 115
1070 LET E=E+1
1100 GOTO 36
1200 INPUT Z$( TO 5)
1202 LET Y=1
1205 FOR X=1 TO 5
1210 IF Z$="      " OR CODE Z$(X)
 AND (CODE Z$(X)<28 OR CODE Z$(X
)>37) THEN LET Y=2
1220 NEXT X
1225 IF Y=2 THEN GOTO 1200
1230 RETURN
2010 PRINT AT 13,0;Q$;Q$;"1ST CO
PY LINE?"
2020 GOSUB 1200
2025 IF VAL Z$>12 THEN GOTO 2020
2030 LET C1=2+VAL Z$
2035 PRINT AT 15,0;"NO.";TAB 13;
"S?"
2045 GOSUB 1200
2047 IF VAL Z$>15-C1 THEN GOTO 2
045
2050 LET C2=VAL Z$
2060 PRINT AT 15,0;" form spacin
g? "
2070 GOSUB 1200
```

NOTES

Then you go back to the Main Menu (line 17)
AUTO SEARCH begins here

C=1 when print outs are desired

Search command goes into X$

If you don't type a number, you start searching at line 27. Otherwise GOSUB 1202
The number input into Z$ goes into E and L.

Then you input an end number
and go to 30 to search
Lines 1008 to 1060 check the LAST WORD of found files in AUTO SEARCHES.
USR 16809 puts the last word into Y$.

If it equals E then the file gets printed

otherwise you go back to search more
1070 and 1100 increase E to the next value to search for
This subroutine checks for valid entry of a number into Z$.

Here is where you "DEFP"

C1=starting line to copy

C2=number of lines to copy

```
2090 LET S=VAL Z$
2100 GOTO 17
3000 PRINT AT C1,0;
3010 POKE 16679,C2
3020 RAND USR 16674
3025 LET Y$="C"
3030 IF C AND B>1 THEN LET Y$=""
3035 FOR X=1 TO S
3036 LPRINT
3037 NEXT X
3040 GOTO 232
```

NOTES

S=number of lines to feed in printing

Printer output starts here

COPY routine

line feeds between forms

then go back to Display Options

VARIABLES IN ORDER OF APPEARANCE

D$ an array of 11000 characters where all data is held
P the number of data bytes holding information
C1 the starting line to COPY to the printer
C2 the number of file lines to COPY
S the number of spaces between forms
Y two uses: a flag to signify AUTO SEARCH and
the line number at which the EDIT cursor blinks
C printer flag (printer output when C=1)
X$ the Search Command Word
B the number of data bytes still to search by USR 16546
E the number currently being searched in ordered file displays
Y$ Used in the display options. Also used as the last word of files in ordered file displays
L the address of the first byte of D$
also used to store the first number for ordered displays
Q$ 32 blank spaces. Used to erase lines of the TV screen
Z$ A number used in AUTO SEARCH and DEFP
X a FOR/NEXT loop to force line feeds to the printer.

INV £ = 140D, 078C H

APPENDIX II The Machine Code

If you already know how to write machine language programs, you should have no trouble understanding the listings that follow. You can skip this text and jump right to the tables.

If you don't know how to communicate directly with your computer's CPU, read on. Perhaps this will be the explanation that makes everything fall into place.

A PRIMER

How many times have you read that programming in machine language is simple or easy or that anyone can do it? I disagree. For the uninitiated, machine language is next to impossible to understand, but once certain fundamentals are grasped, machine code programming can be done by anyone who is patient and able to learn from his mistakes. Is it simple or easy? No. But it is very fast, effieient, and it's exhilarating to watch your program whizz through its paces: like a finely tuned machine with every part working in harmony. It's like watching music.

Don't try to use reason or intellect when you first start, it will only serve to confuse you. When a child learns to speak English he doesn't waste time wondering why Daddy chose to mechanically place his tongue on the roof of his mouth, then lower it, purse his lips, and all the while make a very loud and frightening noise in his throat. What concerns the child is that when Dad makes this "NO!" sound, it means stop what you're doing and watch your step.

When you learn machine language, be like the child. Accept principles on faith. Observe results. Say to yourself: OK, these are the rules. Don't ask: Why do it that way? The computer will not reason with you.

Memory and Organization-Blind Faith Principle No. 1

Imagine a little man in your computer whose job it is to carry out a numbered list of instructions (a program). At his disposal is a vast set of message slots or pigeonholes (memory) that he can put messages into or retrieve messages from. The pigeonholes are neatly arranged so that it takes almost no time at all to pull a message from slot #6 or deposit one in slot #14.

Our man has several work tables (registers) on which he can place messages or just the pigeonhole numbers (addresses) while he is at work. The man's list of instructions are placed in sequential message slots and a special work table tells him which slot holds his next instruction (program counter).

Our man is not very bright. He knows how to add and subtract, but his real strength lies in the fact that he is very good at following directions, and he can do it quickly.

This is the essence of a computer. The man is the CPU inside. You, the programmer, are the boss. You supply the instructions, but you must speak his language. This is where the confusion begins becaus the man doesn't know anything but numbers. To make matters worse, his method of counting is different than yours.

Computer Numbers and How to Use Them-Blind Faith Principle No. 2

A machine language program is a list of numbers. More precisely, it is a numbered list of numbers. On the tables that follow, the number in the first column is the address or pigeonhole # where the instruction in the second column can be found. It's the second number that is the actual instruction.

The numbers are written in hexadecimal or base 16. This is the number system the CPU will understand. But here lies a problem. When the computer asks you for a number or when you ask it for one, the little man inside assumes that you will only want decimal numbers so that's what you'll get.

Therefore, you the programmer must think in 2 number systems. When you deal with numbers you must provide decimal equivalents of hexadecimals. This is what makes machine coding so confusing. Not only do you have to know what effect a hex number has on the computer instruction-wise, but you also have to mentally convert that number into decimal before you place it into the computer's memory.

Placing Numbers in Memory

The owners manual does a fairly good job of explaining how numbers are POKEd into memory and how to PEEK the values stored in various bytes. Whenever you use POKE or PEEK you must specify which pigeonhole or address you mean. Each byte is numbered with its own personal address. The number that any one byte can hold ranges from 0-255 in decimal notation; 00-FF in hex. When you tell the computer to:

```
PRINT PEEK 16725
```

you are telling the computer to display the message (or the value) it finds in pigeonhole #16725.

At the back of the manual, in the appendix called "The Character Set", there is a handy guide to decimal numbers and their hex equivalents. If you want to look up the hex value for the number 64, just find this number in the column called "code" and read the hex number (40) in the column labeled "hex".

At this point you may ask: Why bother with all this fuss over hex and decimal conversions? Since PEEKS and POKES are all in decimal anyway, wouldn't it be easier to avoid hex altogether?

ANSWER: As long as you deal with one byte at a time, yes. It would be

much simpler to just forget that hex ever existed. Unfortunately, machine code must often deal with numbers greater than 255. Since this is the maximum allowed for any one byte, the computer must use 2 bytes to represent numbers of any size. This is where counting in hex begins to shine despite the need to convert each byte to its decimal equivalent.

Consider this example:

code	mnenomics
01	LD BC, NN
BF	
08	

When the computer finds the instruction 01 it means that the man inside is to place the single number it finds in the next two bytes into the BC register pair. In this example the number is 08BF-hex. Notice that the 2 numbers after 01 are in reverse order of the actual number. BC gets loaded with the second byte, then the first.

To expand this into a simple machine language program, the command and its accompanying 2 byte number needs 3 consecutive bytes of memory in which to place the code. Also, a bona fide machine code routine needs a RETURN command at the end, so in all we need 4 bytes.

An excellent place to put a program like this is inside a Basic REM statement. Enter this:

1 REM _ _ _ _ (underlines denote spaces)

The four spaces after the REM you can now POKE to hold the code. With the Sinclair computer, the address of the first space after the word REM in the first line of a program is 16514 (4082 hex). . . always.

Therefore, to POKE the code into REM, use the decimal/hex conversion table to translate the code into its decimal equivalent:

01 hex = 01 decimal
BF hex = 191 decimal
08 hex = 08 decimal

Byte #4, the return command, (C9 in hex) is 201 in decimal. Beginning with address 16514, POKE in your conversions:

```
POKE 16514,1
POKE 16515,191
POKE 16516,8
POKE 16517,201
```

This program doesn't do much, but it illustrates an important point. The BC register pair is loaded with a number: 08BF. The way Sinclair Basic implements this routine is with the USR function: USR 16514. This function does more than just jump to a machine code routine beginning at the address you specify (16514 in this case). It also creates a number. The number is whatever happened to be in the BC register pair when the machine code returns

to Basic. Sometimes it is useful to take advantage of this number. You could, for example, say:

```
LET C=USR 16514
```

Doing this causes the computer to essentially GOSUB to the machine code and also initialize a Basic variable, C, whose value is the BC pair on the machine code's RETURN.

Try this exercise:

```
1 REM
10 PRINT USR 16514
```

POKE the code into the first REM line and run the program. The 2 byte hex number that you poked into addresses 16515 and 16516 has its decimal equivalent printed on the screen. Try poking different numbers into 16515 and 16516, and attempt to predict the outcome of PRINT USR 16514.

You can use Basic to arrive at the answer by typing:

```
PRINT PEEK 16515+256*PEEK 16516
```

When you understand a simple machine code routine to the extent of being able to predict the answer, you are well on your way to becoming an expert at machine code. All that's left to do is learn what the machine language instructions are and what they do. That's the easy part. Go to your nearest Radio Shack and pick up a copy of PROGRAMMING THE Z80 by Rodnay Zaks. Skip the first section on Binary arithmetic and go right to the chapter on the Z80 instruction set. Try using different instructions to produce simple outcomes. Remember to leave enough blank spaces in your first REM line to hold your code. It's better to have too much rather than not enough. Remember, too, that when you use machine code, you work with just one memory byte at a time; in Basic you can handle blocks of memory.

```
40A2  2A  LD HL,(40D6)
40A3  D6
40A4  40
40A5  ED  LD BC,(40D8)
40A6  4B
40A7  D8
40A8  40
40A9  11  LD DE,4082
40AA  82
40AB  40
40AC  1A  LD A,(DE)
40AD  ED  CPIR
40AE  B1
40AF  E2  JP PO,40DA
40B0  DA
40B1  40
40B2  13  INC DE
40B3  1A  LD A,(DE)
40B4  FE  CP 9B
40B5  9B
40B6  CA  JP Z,40C1
40B7  C1
40B8  40
40B9  ED  CPI
40BA  A1
40BB  CA  JP Z,40B2
40BC  B2
40BD  40
40BE  C3  JP 40A9
40BF  A9
40C0  40
40C1  22  LD (40D6),HL
40C2  D6
40C3  40
40C4  2B  DEC HL
40C5  7E  LD A,(HL)
40C6  FE  CP 17
40C7  17
40C8  CA  JP Z,40CE
40C9  CE
40CA  40
40CB  C3  JP 40C4
40CC  C4
40CD  40
40CE  22  LD (407B),HL
40CF  7B
40D0  40
40D1  ED  LD (40D8),BC
40D2  43
40D3  D8
40D4  40
40D5  C9  RET
40D6  00  NOP
40D7  00  NOP
40D8  00  NOP
40D9  00  NOP
40DA  CD  CALL 4119
40DB  19
40DC  41
40DD  01  LD BC,1
40DE  01
40DF  00
40E0  C3  JP 40CE
40E1  CE
40E2  40
40E3  00  NOP

4119  2A  LD HL,(4010)
411A  10
411B  40
411C  01  LD BC,6
411D  06
411E  00
411F  ED  ADC HL,BC
4120  4A
4121  C9  RET
```

The Search Routine, USR 16546, jumps to the instructions beginning at 40A2 hex. Two variables (40D6 and 40D8) store the starting byte to search from and the number of bytes to check respectively. A "found file" has its starting address placed in byte 407B. If the search is complete and no file is found, BC is loaded with 1, and 407B marks the address of the first file: "SEARCH IS COMPLETE". Note that in the case of found files, the routine loads 40D6 and 40D8 with the current location where the file was found so that continuing the search simply picks up where the computer left off before it stopped to display a file. The subroutine at 4119 loads HL with the 6th byte after VARS: the first data byte.

```
40E9  ED  LD BC,(400E)
40EA  4B
40EB  0E
40EC  40
40ED  C9  RET
```

40E9 hex=USR 16617 This routine checks the first character of a line to be Added. The code for the character found is the result on return to Basic.

```
40F0  ED  LD DE,(407B)
40F1  5B
40F2  7B
40F3  40
40F4  01  LD BC,0
40F5  00
40F6  00
40F7  D5  PUSH DE
40F8  13  INC DE
40F9  03  INC BC
40FA  1A  LD A,(DE)
40FB  FE  CP 17
40FC  17
40FD  C2  JP NZ,40F8
40FE  F8
40FF  40
4100  D5  PUSH DE
4101  ED  LD (407B),BC
4102  43
4103  7B
4104  40
4105  01  LD BC,0
4106  00
4107  00
4108  03  INC BC
4109  13  INC DE
410A  1A  LD A,(DE)
410B  FE  CP 86
410C  86
410D  C2  JP NZ,4108
410E  08
410F  41
4110  E1  POP HL
4111  D1  POP DE
4112  ED  LDIR
4113  B0
4114  ED  LD BC,(407B)
4115  4B
4116  7B
4117  40
4118  C9  RET
```

40F0 hex=USR 16624 The file deletion. When USR 16624 is called, the file to be deleted has its starting address stored in FILEPEEK (407B). This address is loaded into DE. The BC pair is set to 0. Then the routine steps ahead until it finds a 17 (the code for "*") marking the next file. This address is stacked. The number of bytes is stored in FILEPEEK. The number of bytes to the end of data is counted.

The LDIR instruction at 4112 moves the block of files beginning with next file and running to the end, up the number of bytes the file to be deleted contains.

The old file is effectively overwritten by all the remaining files. The number of bytes the old file occupied is loaded into BC so that the Basic can subtract this number from the variable P at line 300.

Read more about this routine on page 35.

```
4122  2A  LD HL,(400E)
4123  0E
4124  40
4125  C5  PUSH BC
4126  16  LD D,B
4127  0B
4128  C3  JP 87E
4129  7E
412A  08
```

4122 hex=USR 16674 the printer routine. Here HL is set to the starting line to copy. The number of lines to print is poked into address 4127 so the code loads this data into D. Then the program jumps into the middle of the ROM Copy routine at 87E.

```
4130  ED  LD BC,(407B)
4131  4B
4132  7B
4133  40
4134  03  INC BC
4135  0A  LD A,(BC)
4136  ED  LD (407B).BC
4137  43
4138  7B
4139  40
413A  FE  CP 17
413B  17
413C  CA  JP Z,4146
413D  46
413E  41
413F  FE  CP 8C
4140  8C
4141  C8  RET Z
4142  D7  RST 10 H
4143  C3  JP 4130
4144  30
4145  41
4146  0B  DEC BC
4147  0A  LD A,(BC)
4148  FE  CP 17
4149  17
414A  C2  JP NZ,4146
414B  46
414C  41
414D  ED  LD (407B),BC
414E  43
414F  7B
4150  40
4151  C9  RET
```

4130 hex=USR16688 Screen File Display The found file listed in FILEPEEK (407B) is taken one character at a time and placed in the Accumulator. The character is compared with 17 (a "*") and 8C (inverse pound). These characters will force a return to Basic with the BC pair loaded with the address of the character that forced the return. Any other character, the address of which is in BC and the character itself in A, has RST 10 applied to it. This is a ROM subroutine that prints whatever is in the Accumulator. After printing, the routine jumps back to test and print the next character.

```
4160  00  NOP
4161  00  NOP
4162  ED  LD DE,(4160)
4163  5B
4164  60
4165  41
4166  01  LD BC,0
4167  00
4168  00
4169  ED  LD HL,(400E)
416A  6B
416B  0E
416C  40
416D  E5  PUSH HL
416E  03  INC BC
416F  23  INC HL
4170  7E  LD A,(HL)
4171  FE  CP 76
4172  76
4173  C2  JP NZ,416E
4174  6E
4175  41
4176  2B  DEC HL
4177  0B  DEC BC
4178  7E  LD A,(HL)
4179  FE  CP 0
417A  00
417B  CA  JP Z,4176
417C  76
417D  41
417E  23  INC HL
417F  36  LD (HL),8C
4180  8C
4181  E1  POP HL
4182  03  INC BC
4183  03  INC BC
4184  C5  PUSH BC
4185  ED  LDIR
4186  B0
4187  C1  POP BC
4188  C9  RET
```

4160 & 4161 are a 2-byte variable (P_PTR)

4162 hex=USR 16738, the file adding routine. The DE register pair is loaded with the last bytes used in the D$ array. BC is zeroed. HL receives the address of the print position of the Display File. (system variable FD_CC).

The content of this address is checked for 76 hex (118 decimal) signifying the end of a line of display. Then the routine steps back through the line checking for spaces. These are the blank spaces after a line of text. When the computer finds a character instead of a space it loads the byte after with CHR$ 8C (inverse pound).

Then, as in deleting, LDIR is used to block load the line of text into the next free spaces of D$. BC is given the number of characters the line of text used so that on return, the Basic can add this number to the variable P:

LET P=P+USR 16738

This routine works on just one line of the Display file at a time. The Loop implemented in Basic takes each line in turn.

```
418C  01  LD BC,0
418D  00
418E  00
418F  ED  LD HL,(407B)
4190  6B
4191  7B
4192  40
4193  11  LD DE,4092
4194  92
4195  40
4196  23  INC HL
4197  3E  LD A,17
4198  17
4199  BE  CP (HL)
419A  C8  RET Z
419B  1A  LD A,(DE)
419C  BE  CP (HL)
419D  C2  JP NZ,4193
419E  93
419F  41
41A0  13  INC DE
41A1  1A  LD A,(DE)
41A2  FE  CP 9B
41A3  9B
41A4  C2  JP NZ,4196
41A5  96
41A6  41
41A7  01  LD BC,1
41A8  01
41A9  00
41AA  C9  RET
```

418C hex= USR 16780, The second word search. The address of the found file (if a match for the first word is found) is loaded from FILE PEI into HL. DE has the address of the first charact of the second word (4092) loaded into it. The Accumulator is loaded with 17 (a "*"). Each character of the file is then checked for a "*" indicating the beginning of the next file. When found, the routine returns to Basic. The BC pair was initialized to a value of zero. Begir ning at address 419B, the Accumulator takes on the value of the first character of the second word buffer. Each character of the file is check against the buffer character. If they don't matcl the operation repeats until the next file (*) is encountered and a return is made. In this case the number resulting from the USR call is zero.

If a match is found the next buffer character is checked. Matches repeat in this manner until 9B (inverse period) is found. This means that the second word is found. BC is given the value of 1 and a return is made.

The second word search will result in one of two possible values: 0 if the file does not contain a match; 1 if it does contain a match.

```
41B4  ED  LD HL,(400C)
41B5  6B
41B6  0C
41B7  40
41B8  23  INC HL
41B9  11  LD DE,4082
41BA  82
41BB  40
41BC  ED  LD BC,(402E)
41BD  4B
41BE  2E
41BF  40
41C0  7E  LD A,(HL)
41C1  FE  CP 18
41C2  18
41C3  CC  CALL Z,41DC
41C4  DC
41C5  41
41C6  ED  LDI
41C7  A0
41C8  EA  JP PE,41C0
41C9  C0
41CA  41
41CB  CD  CALL 4119
41CC  19
41CD  41
41CE  22  LD (40D6),HL
41CF  D6
41D0  40
41D1  00  NOP
41D2  00  NOP
41D3  00  NOP
41D4  00  NOP
41D5  00  NOP
41D6  3E  LD A,9B
41D7  9B
41D8  12  LD (DE),A
41D9  C9  RET
41DA  00  NOP
41DB  00  NOP
41DC  3E  LD A,9B
41DD  9B
41DE  12  LD (DE),A
41DF  11  LD DE,4091
41E0  91
41E1  40
41E2  12  LD (DE),A
41E3  13  INC DE
41E4  23  INC HL
41E5  0B  DEC BC
41E6  79  LD A,C
41E7  D6  SUB E
41E8  0E
41E9  3E  LD A,9B
41EA  9B
41EB  32  LD (41F3),A
41EC  F3
41ED  41
41EE  F8  RET M
41EF  01  LD BC,E
41F0  0E
41F1  00
41F2  C9  RET
41F3  00  NOP
```

41B4 hex=USR 16820, the X$ buffer loader. Here, the HL pair is loaded with the address of the Display File. DE is given the address of the first byte of the buffer (4082 hex, or 16514 decimal). BC is given the value held in 402E (STRLEN). In turn, each character from HL (where the search word is obtained) is checked for a "/". If one is found the program calls a subroutine beginning at 41DC. Finding a "/" means that there are two words to put in the buffer. First an inverse period is placed at the end of word 1. Then DE is given the starting address of the second word buffer (4091).

Placing the word into the buffer is accomplished with the LDI command at address 41C6. Each character is placed into successive bytes of the buffer. The Jump instruction repeats the loading until the last character of X$ has been loaded. Then subroutine 4119 loads the HL pair with the address of the first file, This address is then loaded into D_PTR in anticipation of the subsequent search. CHR$ 9B (the inverse period) is finally placed in the last position of the search buffer.

```
41FA  ED  LD HL,(407B)
41FB  6B
41FC  7B
41FD  40
41FE  3E  LD A,17
41FF  17
4200  23  INC HL
4201  BE  CP (HL)
4202  20  JR NZ,FC (4200)
4203  FC
4204  2B  DEC HL
4205  01  LD BC,5
4206  05
4207  00
4208  3E  LD A,0
4209  00
420A  ED  CPDR
420B  B9
420C  E5  PUSH HL
420D  21  LD HL,5
420E  05
420F  00
4210  ED  SBC HL,BC
4211  42
4212  E5  PUSH HL
4213  C1  POP BC
4214  ED  LD HL,(402E)
4215  6B
4216  2E
4217  40
4218  ED  LD DE,(4012)
4219  5B
421A  12
421B  40
421C  ED  ADC HL,DE
421D  5A
421E  23  INC HL
421F  23  INC HL
4220  23  INC HL
4221  EB  EX DE,HL
4222  E1  POP HL
4223  23  INC HL
4224  ED  LDIR
4225  B0
4226  C9  RET
```

41FA=USR 16890. This is the routine that works in the AUTOSEARCH mode to put the last word of a file into Y$.

First, HL is loaded with the address held in FILEPEEK. The Accumulator is given the value 17 (*). Then the 3 instructions beginning at 4200 step through the file to the end. At this point HL holds the address of the last character. BC is then set to 5, the accumulator set to 0 (a space). The CPDR instruction next checks the last 5 characters of the file or until it finds a space, leaving the address in HL and the number of characters checked in BC. The routine counts back 5 spaces or until it comes to a space--whichever is less. The actual number of bytes checked is subtracted from 5 and put into BC.

HL is next given the value stored in STRLEN DE is loaded with the address given in system ·able DEST (4012 hex). The length of the lasi ord (or the last 5 characters) stored in DE is then added to the address in HL. The sum has 3 added to it, and then it is placed in the DE pair. This manipulation causes the DE pair to point to the destination (Y$) where the last file word is to be sent. The address of the file word is popped from the stack into HL, and the LDIR instruction loads the characters.

Read the text on page 34 for a graphic illustration of what this routine does.

There's More. . . . .

"PRO/FILE Updates" is a quarterly newsletter that brings you even more good stuff about ZX PRO/FILE. "Updates" contains reports by users who find unusual and particularly useful applications, reviews of hardware that add to the program's usefulness, and modifications for adapting PRO/FILE to such things as stringy-floppies, disks, megabytes of RAM.

With "PRO/FILE Updates" you can keep your program from going out of date. You'll get instructions for adding new data processing capabilities like automatic alphabetizing, fast zip code sorting, upgrading to larger memories without losing previously entered files, and much more.

Price: $9.95 per year. Issues appear in January, April, July, and October.

Thomas B. Woods, P.O. Box 64, Jefferson, NH 03583

Dear Tom,

Send me "PRO/FILE Updates". Enclosed is $9.95 for a 1 year's subscription.

Name_____

Street_____

City/State/Zip_____